commodore 16
ESSENZIALI E ROUTINE

David Lawrence

GRUPPO EDITORIALE JACKSON

commodore 16
ESSENZIALI E ROUTINE

David Lawrence

GRUPPO
EDITORIALE
JACKSON
Via Rosellini 12
Milano

Copyright per l'edizione originale:
Sunshine Books - 1984
Titolo originale: THE WORKING COMMODORE C16

Copyright per l'edizione italiana:
GRUPPO EDITORIALE JACKSON - Aprile 1985

TRADUZIONE: Daniela Giacomelli
GRAFICA E IMPAGINAZIONE: Marco Passoni
SUPERVISIONE TECNICA: Vittorio Riva
COPERTINA: Silvana Corbelli
FOTOCOMPOSIZIONE: SYSTEM GRAPHIC - Cologno Monzese (MI)
STAMPA: Grafika '78 - Pioltello (MI)

Tutti i diritti sono riservati. Stampato in Italia. Nessuna parte di questo libro può essere riprodotta, memorizzata in sistemi di archivio, o trasmessa in qualsiasi forma o mezzo, elettronico, meccanico, fotocopia, registrazione o altri senza la preventiva autorizzazione scritta dell'editore.

INDICE

PREFAZIONE ..... » VII
Come usare il libro ..... » IX

CAPITOLO 1. Esperimenti con l'ora ..... » 1
— Orologio analogico ..... » 1
— Orologio ..... » 15
— Timer ..... » 21
— Dayplan ..... » 37

CAPITOLO 2. Dipingere con i numeri ..... » 51
— Graph ..... » 51
— Pie Chart ..... » 59
— Three Dimensional Graph ..... » 65

CAPITOLO 3. Suoni e luci ..... » 77
— Artist ..... » 77
— Display ..... » 99
— Characters ..... » 101
— HDraw ..... » 117
— Music ..... » 124

CAPITOLO 4. Sempre più seriamente ..... » 137
— Unifile ..... » 137
— Nnumber ..... » 154
— Texted ..... » 167

CAPITOLO 5. Questioni di soldi ..... » 185
— Banker ..... » 185
— Accountant ..... » 195

IL CONTENUTO IN DETTAGLIO

CAPITOLO 1 — **Esperimenti con l'ora**
OROLOGIO ANALOGICO crea e fa funzionare un orologio completo di quadrante e lancette. OROLOGIO II offre all'utente la possibilità di riprodurre lo scorrere del tempo in un modo nuovo e originale. TIMER crea dieci timer in grado di funzionare simultaneamente e di emettere un segnale acustico (più un promemoria.) E di generare un messaggio d'avvertimento. DAYPLAN è un ampliamento del programma TIMER che permette di memorizzare e visualizzare gli appuntamenti di sette giorni consecutivi.

CAPITOLO 2 — **Dipingere con i numeri**
GRAPH crea un grafico linerare ad alta risoluzione. PIECHART prende pochi dati e li visualizza sotto forma di diagramma a torta suddiviso in settori multicolori. GRAPH II permette di creare un diagramma a barre tridimensionali.

CAPITOLO 3 — **Suoni e luci**
ARTIST permette di realizzare disegni a bassa risoluzione. DISPLAY visualizza rapidamente una selezione di immagini create sullo schermo con il programma precedente. CHARACTERS consente la creazione di caratteri personalizzati. HDRAW è un sofisticato strumento per la creazione di disegni ad alta risoluzione. MUSIC permette di elaborare e riascoltare motivi a due voci.

CAPITOLO 4 — **Sempre più seriamente**
UNIFILE è un efficace sistema di archiviazione in grado di memorizzare una vasta gamma di informazioni, recuperabili rapidamente in qualsiasi momento. NNUMBER è un programma che crea un dizionario di nomi e numeri per qualsiasi tipo di informazioni e di impiego, consentendo all'utente di elaborare fatture, listini di borsa e persino il calcolo delle calorie della propria dieta quotidiana. TEXTED è un word processor molto semplice in grado di lavorare in ambito BASIC.

CAPITOLO 5 — **Questioni di soldi**
BANKER permette all'utente di tenere una registrazione personale, chiara e sempre aggiornata del proprio conto corrente bancario. ACCOUNTANT è in grado di gestire una piccola serie di conti, suddivisi in debiti e crediti e riprodotti nella loro forma consueta.

PREFAZIONE

Questo libro fa parte di una delle serie di pubblicazioni per microcomputer che hanno incontrato maggiore successo. Nel 1982, quando venne pubblicato il primo libro della serie Working Micro, vi erano ancora ben pochi editori convinti che i possessori di microcomputer fossero desiderosi di usare e capire le proprie macchine e di acquisirne il controllo, imparando a programmare... programmando. A quel tempo la maggior parte dei libri proponevano giochi e passatempi oppure si configuravano come guide per principianti. La prospettiva di un libro che fornisse una buona raccolta di programmi utili e seri, e allo stesso tempo una descrizione abbastanza dettagliata dei metodi di programmazione usati da utenti più esperti, non aveva ancora incontrato l'interesse degli addetti ai lavori.
Già allora ero personalmente convinto, e con me tutti i collaboratori della Sunshine Books, che i libri della serie Working Micro rappresentassero esattamente ciò che il pubblico stava aspettando e che ci sarebbe stata ben presto una grossa carenza nell'offerta di libri per quello che sarebbe diventato un esercito di possessori di microcomputer. Avevamo ragione.
Da allora i libri della serie Working Micro hanno seguito i microcomputer in tutti i mercati in cui sono riusciti a penetrare e a diffondersi. Attualmente sono tradotti in 14 lingue e il successo più grande finora è stato il volume dal titolo "The Working Commodore 64". Da quando è stato scritto e da quando sempre più persone hanno scoperto le qualità del 64, è stato dimostrato che l'approccio seguito dal libro e la potenza della macchina sono stati fatti l'uno per l'altra.
Ora stanno arrivando i computer Commodore della nuova generazione, fra cui il C16, non così potente come il Commodore 64, in termini di memoria, ma molto più avanti per quanto riguarda il linguaggio BASIC che utilizza. Alla base di questo libro, un solo impegno: provare a prendere sul serio le capacità di una macchina a 16K. Indubbiamente saranno in molti a dire che non si può fare niente di serio con una memoria di queste dimensioni, ma quando sarete arrivati alla fine del libro vi renderete conto di quanto abbiano torto e di come sia divertente provarlo.
Questo libro può essere usato in molti modi:
1) come raccolta di programmi che possono essere adattati e sviluppati per i propri scopi
2) come insieme di subroutine con le quali è possibile costruire i propri programmi
3) come introduzione alla programmazione con il BASIC C16.
Comunque decidiate di usarlo, ricordate che è stato scritto come un libro e non come una raccolta di programmi a caso. Molto spesso mi è capitato di incontrare lettori alle prese con problemi insormontabili perché nel mezzo di programmi

molto complessi senza una preparazione adeguata.
I primi programmi del libro, benché utili ed interessanti in sé, sono stati inseriti come introduzione a ciò che verrà sviluppato in seguito. Affiancate ai programmi, sono state inserite spiegazioni di ottimo livello in modo che, nel momento in cui vi troverete ad affrontare problemi più complessi, sarete in grado di capirne a fondo le tecniche utilizzate.
Nello scrivere questo libro ho potuto contare sull'aiuto della Commodore UK che ha messo a mia disposizione macchine e consigli. Un ringraziamento particolare va soprattutto a Gail Wellington. Il periodo che precede il lancio di un nuovo prodotto non è il migliore per rispondere a domande o per risolvere problemi altrui. Il fatto che il libro sia qui significa che il tempo per farlo è stato trovato.

COME USARE IL LIBRO

Importante: I programmi contenuti in questo libro sono stati tutti elaborati in modo tale da rendere i caratteri di controllo, che essi contengono, più facilmente riconoscibili. In tutto il libro, essi sono stati sostituiti da frasi racchiuse fra parentesi quadre. Poiché non esistono occasioni in cui le parentesi quadre siano usate per altri scopi, non ci saranno dubbi che esse indichino sempre caratteri di controllo. Ecco dunque qui di seguito la tabella delle rappresentazioni fra parentesi quadre e i caratteri di controllo corrispondenti:

CURSORE:	SU	—	CU
	GIÚ	—	CD
	SINISTRA	—	CL
	DESTRA	—	CR
CANCELLA		—	DEL
INSERIRISCI		—	INST
AZZERA SCHERMO		—	CLEAR
POSIZIONE RIPOSO CURSORE		—	HOME
INVERSIONE ON		—	RVS ON
INVERSIONE OFF		—	RVS OFF
LAMPEGGIAMENTO ON		—	FLASH ON
LAMPEGGIAMENTO OFF		—	FLASH OFF
NERO		—	BLK
BIANCO		—	WHT
ROSSO		—	RED
AZZURRO		—	CYN
PORPORA		—	PUR
VERDE		—	GRN
BLU		—	BLU
GIALLO		—	YEL
ARANCIONE		—	ORNG
MARRONE		—	BRN
GIALLO-VERDE		—	YL GRN
ROSA		—	PINK
VERDE-BLU		—	BL GRN
AZZURRO CHIARO		—	L BLU
BLU SCURO		—	D BLU
VERDE CHIARO		—	L GRN

CAPITOLO 1

ESPERIMENTI CON L'ORA

È sempre difficile sapere da dove incominciare con un libro come questo. Se i programmi proposti sono troppo compelssi, il lettore si troverà confuso prima di aver acquisito quelle nozioni che possono aiutarlo a rendergli i programmi sempre più facili da capire a mano a mano che procede con la lettura. Se invece i primi programmi sono troppo facili, non si sente spinto ad andare avanti, convinto che non ci sia niente di più sostanziale da imparare.
Abbiamo così deciso di fermarci, in questo primo capitolo, a quattro programmi che hanno a che fare con l'ora e con i modi in cui essa può essere trattata sul C16. I programmi sono relativamente semplici, ma ci permettono di presentare una serie nutrita di concetti che verranno ripresi in seguito in programmi più complessi. Inoltre, per il loro impiego dei calcoli, del suono e della grafica a bassa e ad alta risoluzione, i programmi rappresentano una buona dimostrazione di alcune delle prestazioni più interessanti del C16.

I quattro programmi che stiamo per proporvi sono:

OROLOGIO ANALOGICO, che produce ad alta risoluzione il quadrante di un orologio.

CLOCK, che fornisce un modo tutto particolare di dire l'ora.

TIMER, che genera dieci timer in funzione simultaneamente, ciascuno in grado di emettere un segnale acustico e di generare un messaggio di avvertimento.

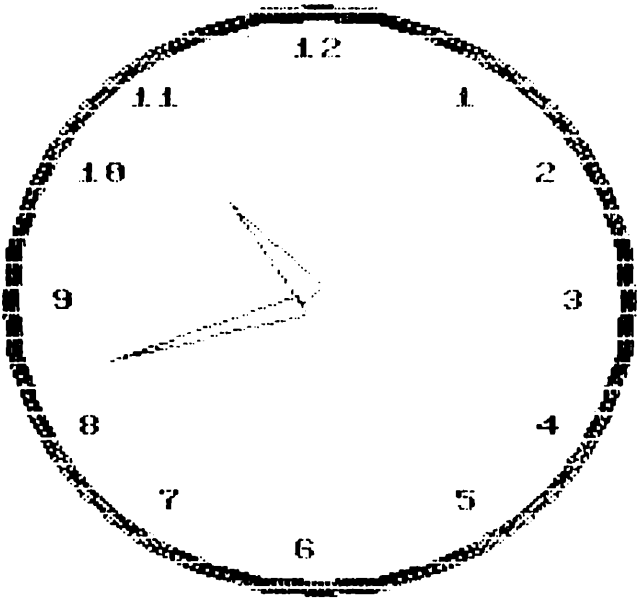
DAYPLAN, un ampliamento del programma TIMER che permette di memorizzare e visualizzare un'agenda per sette giorni.

Programma 1.1: Orologio analogico

Scopi del programma

Scopo di questo programma è riprodurre su schermo il quadrante di un orologio completo di lancette che si muovono con il trascorrere dei minuti, istruzioni dopo istruzione. Seguendo il programma, imparerete molto sui metodi adottati nel libro, purché leggiate attentamente tutti i commenti.

Figura 1.1



Nel programma sono stati introdotti i seguenti concetti:
1) Salvataggio del programma durante lo sviluppo
2) Inizializzazione delle variabili importanti
3) Funzioni definite
4) Moduli di controllo
5) Uso dei cicli DO
6) Comando TRAP
7) Modo ad alta risoluzione (GRAPHIC 1)
8) Flusso logico e leggibilità del programma
9) Programmazione modulare

Modulo 1.1.1: Salvataggio del programma

(Importante: Prima di procedere con questo o con qualsiasi altro programma, leggete la nota sulla presentazione dei programmi, che precede il Capitolo 1.
Queste tre linee possono sembrare banali per incominciare, ma quelli che hanno lavorato con i nostri libri per altri modelli sanno che questo piccolo modulo può far risparmiare un mucchio di grattacapi nello sviluppo dei programmi.
La maggior parte della gente impara a proprie spese che i programmi vanno regolarmente salvati a mano a mano che vengono sviluppati. Prima o poi quasi tutti si trovano a dover correre contro il tempo quando ore e ore di lavoro se ne vanno in fumo per un'interruzione di corrente, un fusibile saltato o un urto subito dal sistema.

In questi casi, gli utenti più esperti si riconosceranno dalla quantità di lavoro perso, che sarà sempre relativamente contenuta, dato che è buona norma non lasciare mai passare più di 15 minuti senza salvare il proprio lavoro.
Scopo delle tre linee di questo modulo è di incoraggiarvi a creare regolarmente copie del programma su cui state lavorando, introducendo semplicemente il comando GOTO 2, e, se state usando un drive a disco, assicurandovi che il programma sia controllato ogni volta che viene salvato, mediante l'uso di VERIFY.
Un'altra funzione importante di questo modulo é che, aggiunto davanti ai programmi, fornisce una linea di partenza standard, cioè 1, permettendo un notevole risparmio di tempo. Se sono state predisposte delle variabili che si vogliono annullare con RUN, è spesso auspicabile iniziare un programma con GOTO: con il modulo collocato in questa posizione, non occorre stabilire il numero della prima linea perché già sapete che potete sempre iniziare con un GOTO 1.
Il modulo precederà sempre tutti i nostri programmi, ma, dato che cambia solo il nome del programma, questa è l'unica volta in cui ne parliamo.

Modulo 1.1.1: Linee 1-3

```
1 GOTO 3
2 SAVE "@0:ANALOGICO.F",8:VERIFY"ANALOGICO.F",
8:STOP
3 REM
```

Listato alternativo per chi usa la cassetta

```
1 GOTO 3
2 SAVE "ANALOGICO": STOP
3 REM
```

Commento:

Linea 2: L'aggiunta di @0: all'inizio del nome del file fa sì che ogni volta che il programma viene salvato (SAVE), vada a scriversi sopra le versioni precedenti. Alcuni programmatori adottano un approccio ancora più cautelativo numerando tutte le versioni del programma (es.: ANALOGICO 01 ANALOGICO 02, ecc). Il vantaggio di una scelta di questo tipo sta' nel fatto che se qualcosa va storta durante il salvataggio del programma, che viene cancellato dalla memoria e non salvato correttamente sul disco, esiste sempre una versione precedente recuperabile.
Non appena il programma è terminato, le versioni di sviluppo possono essere cancellate dal disco.
Gli utenti delle cassette non hanno bisogno di tutto questo dato che, qualsiasi cosa ci sia sul nastro, viene comunque sostituita dal nuovo contenuto. Inoltre, la funzione VERIFY viene omessa nella versione su nastro a causa dell'enorme

quantità di tempo che occorrerebbe in questo caso per eseguire le operazioni di salvataggio (SAVE) e di verifica (VERIFY). Dopo aver effettuato il salvataggio, si consiglia, a questo proposito, di riavvolgere il nastro e di eseguire una verifica (VERIFY) del programma.

Verifica

Controllate che il drive contenga un disco (o che nel registratore ci sia una cassetta), quindi battete:

GOTO 2 [RETURN]

Il comando farà partire il drive e, dopo un attimo, vedrete comparire la scritta BREAK IN 2. A questo punto potrete cancellare le tre linee in memoria e ricaricare il programma da disco, battendo:

NEW[RETURN] (che cancella il programma corrente)
DLOAD "ANALOGICO" [RETURN]

Non appena il processo di caricamento si è concluso, listate il programma e il modulo dovrebbe essere stato ricaricato.
Chi usa il nastro lo riavvolgerà e batterà:

LOAD "ANALOGICO" [RETURN]

Modulo 1.1.2: Inizializzazione delle variabili

Tutti i programmi che meritano questo nome utilizzano variabili e costanti, cioè etichette, i cui valori possono essere modificati in fase di elaborazione del programma o almeno da programma a programma. Il vantaggio rappresentato dalle variabili è di permettere la scrittura di linee di programma che si applicano a più di una situazione, così, ad esempio, PRINT 2*A può essere usato ovunque, indipendentemente dal valore di A.
Sono molto poche le variabili che devono assolutamente avere i propri valori dichiarati quando il programma viene eseguito (RUN) per la prima volta e molti aspettano di arrivare a metà programma, quando la variabile è di importanza vitale, per definirne il valore.
È una politica sbagliata: infatti, a mano a mano che il programma viene sviluppato, risulta sempre più difficile vedere qual è il valore delle variabili importanti quando il programma inizia per la prima volta. In generale, è buona norma dichiarare il valore delle variabili principali fin dall'inizio del programma con una procedura che, per l'appunto, è nota come "inizializzazione".
Detto questo, esiste un'eccezione nel caso in cui la memoria sia limitata. In questo caso, è meglio tralasciare le variabili il cui valore non riveste un ruolo importante nel momento in cui il programma viene eseguito per la prima volta. Più avanti nel programma, vi troverete davanti a variabili chiamate TT, TO, M2, H2, A1, A2 e A3. Il loro valore è irrilevante quando il programma inizia perché

sono derivate da altri valori durante l'esecuzione del programma e vengono usate solo temporaneamente. Aggiungerebbe chiarezza al programma il fatto di elencarle all'inizio in una linea come:

```
270 TT=0:T0=0:M2=0...ecc
```

in modo che chiunque, guardando il programma, sia in grado di vedere esattamente quali variabili sono state usate anche se non stanno a zero per molto tempo. Nel caso di una macchina con 16K di memoria, comunque, un programma come questo lascia ben poco spazio e una linea come questa rappresenta davvero un lusso.

Modulo 1.1.2: Linee 1000-1070

```
1000 REM*********************************
1010 REM INIZIALIZZAZIONE
1020 REM*********************************
1030 DEFFNCM(X)=25*COS((X-90)/180*3.14)
1040 DEFFNSM(X)=25*SIN((X-90)/180*3.14)
1050 DEFFNCH(X)=15*COS((X-90)/180*3.14)
1060 DEFFNSH(X)=15*SIN((X-90)/180*3.14)
1070 COLOR0,10,2:COLOR1,12,6
```

Commento

Il modulo di inizializzazione di OROLOGIO ANALOGICO non viene di fatto usato per predisporre nessuna variabile ordinaria, dato che i soli valori realmente importanti vengono introdotti dall'utente solo in un successivo momento. La sua funzione è quella di dichiarare quattro variabili di un tipo un po' speciale, conosciute come "funzioni definite dall'utente". Una funzione è un miniprogramma incorporato nella ROM del C16, che prende un valore o una stringa specificati dall'utente e vi esegue una semplice operazione per ottenere un altro valore o un'altra stringa.

Prendiamo l'esempio della funzione INT, dove l'istruzione, per ricavare il valore INT (10.5), chiama una routine che stacca il valore .5 dal numero, lasciando un numero intero. Oltre a usufruire di tutte le funzioni incorporate fornite dal C16, l'utente ha la possibilità di definire per sé una funzione o una semplice operazione da eseguire su un numero (non su una stringa).

La definizione di una funzione viene eseguita utilizzando l'istruzione DEFFN con una funzione etichettata definita uguale ai risultati di una formula introdotta dall'utente. Per esempio, DEFFNA (X)=X+Y significa che ogni volta che una funzione di nome A (FNA) viene riscontrata durante l'esecuzione del programma, il suo valore diventa uguale a quello di X+Y in quel momento. Una possibilità in più è data dal fatto che, se la variabile specificata fra parentesi, a sinistra dell'equazione, si trova anche sulla destra, nel calcolo del valore di una funzione, può essere sostituita in modo che FNA (Z) sia uguale a Z+Y, con Z in sostituzione della X originaria.

Definizione di un cerchio

Le funzioni definite in questo modulo particolare saranno usate in seguito per evidenziare certe posizioni intorno ad un cerchio. Esse si basano sul fatto che è sempre possibile stabilire la posizione di un punto della circonferenza, quando si dispone delle seguenti informazioni:

a) Il raggio del cerchio (RADIUS)

b) L'angolo che deve muoversi in senso orario dalle tre fino ad un'ora prestabilita (ANGLE)

c) Le coordinate dal centro del cerchio (CENTRE X e CENTRE Y).

Dati questi tre dati, la posizione sarà espressa da due formule:

coordinata X = RADIUS*COSINE (ANGLE/180*PI)+CENTRE X

coordinata Y = RADIUS*SINE (ANGLE/180*PI)+ CENTRE Y

Lo spazio a nostra disposizione non ci consente di analizzare il perché di queste affermazioni, ma qualsiasi buon trattato di trigonometria vi darà tutte le informazioni di cui potete sentire il bisogno.

Le quattro funzioni definite, incluse nel modulo, verrano usate in seguito per posizionare le lancette del nostro orologio. Il loro scopo è di calcolare le coordinate dei punti di due cerchi, uno con un raggio di 25 unità e l'altro con un raggio di 15.

Modulo 1.1.3: Impostazione dell'ora

Prima di imbarcarci nella creazione del nostro orologio, dobbiamo avere gli strumenti per impostare l'ora. La funzione di questo modulo è di permettere all'utente di introdurre ore e minuti e di memorizzarli nel clock interno permanente del C16. Nel corso dell'esecuzione del modulo, incontreremo una descrizione dell'istruzione TRAP, dell'uso dei cicli DO (DO loop) e della variabile di sistema TI$ che registra l'ora corrente.

Modulo 1.1.3: Linee 6000-6190

```
6000 REM*******************************
6010 REM INSERIMENTO TEMPO
6020 REM*******************************
6030 TRAP6160
6040 COLOR0,10,2
6050 COLOR1,12,6
6060 DO UNTIL RIGHT : RIGHT=1
6070 SCNCLR
6080 CHAR,3,3,"●●●●●●●●●●●●●●●●●●●●●●●●●"
6090 CHAR,9,4,"●  OROLOGIO ANALOGICO  ●"
6100 CHAR,9,5,"●●●●●●●●●●●●●●●●●●●●●●●●●"
```

```
6110 PRINT : INPUT
"[CD][CD][CD][CD][CD][CD]DAMMI LE ORE
(HHMM)";[CD]$
6120 TI$=[CD]$+"00"
6130 LOOP
6140 TRAP
6150 RETURN
6160 PRINT "[CD][CD]ERRORE: DEVI INSERIRE [RVS
ON]HHMM[RVS OFF]-ES.:0945"
6170 FORI=1 TO 3000 : NEXT
6180 RIGHT=0
6190 RESUME NEXT
```

Commento

Linee 6030, 6140 e 6160-6190: Queste linee formano un'unità funzionante che ci permette di fare uso della funzione TRAP. Normalmente, quando il C16 incontra un errore in un programma, la sua esecuzione produce un errore. In caso di errori di sintassi, non interromperemo il processo dato che è essenziale che certi errori siano corretti. Talvolta, però, è possibile prevedere il tipo di errori che potrebbero insorgere in risposta ad un dato non valido, introdotto ad un certo punto durante l'esecuzione del programma ed è qui che TRAP si rivela utile.
La funzione di TRAP è di comunicare al C16: "Se incontri un errore che potrebbe interrompere il programma, salta a questa o a quella linea e continua."
Usando TRAP, è possibile, in certi casi, fare in modo che il C16 lavori con gli errori che altrimenti ci costringerebbero a prendere delle contromisure nell'ambito del programma. Il modulo ci fornisce un esempio utile.
Nell'ambito del ciclo che inizia dalla linea 6060, all'utente viene chiesto di dare il valore alla variabile TI$ che registra l'ora per il C16. Il formato di questa stringa, che è illustrato qui di seguito, deve essere giusto e normalmente va esaminato molto accuratamente prima della accettazione definitiva della stringa; in caso contrario il programma si fermerebbe con un errore ad ogni dato sbagliato introdotto. Usando TRAP, invece, è lo stesso C16 ad individuare i formati errati e a saltare alla linea 6160 dove viene generato un messaggio di errore. L'istruzione di ripresa alla linea 6190 dice al C16 di riprendere l'esecuzione del programma dalla linea immediatamente successiva a quella (6120) che ha generato l'errore. Infine, non appena l'ora è stata accettata, l'istruzione TRAP senza relativo numero di linea, alla linea 6140, riporta il C16 nel suo stato normale, cioè quello in cui l'errore ferma l'esecuzione del programma. Questo perché non vogliamo che altri errori del programma provochino un salto indietro verso questo modulo.
Una nota di avvertimento va inserita a questo punto sull'uso di TRAP durante lo sviluppo del programma. TRAP non distingue fra i diversi tipi di errore, così se

erroneamente introducete il programma in modo che generi un errore che non avete anticipato, predisponendo TRAP in modo che salti ad un certo numero di linea, farete sì che non possiate vedere il messaggio di errore. Se ad esempio avete introdotto erroneamente la linea 6060 in modo che normalmente genererebbe un errore di sintassi, con TRAP attivato, il programma salta alla linea 6160. Se vi trovate sempre ad una linea che contiene un messaggio di errore da voi attivato e non riuscite a capire perché, vi consigliamo di togliere temporaneamente l'istruzione TRAP in modo da essere sicuri che non nasconda un messaggio di errore di importanza vitale. I programmi che vedremo in seguito ci permetteranno di vedere in che modo è possibile ovviare all'inconveniente senza togliere l'istruzione.

Linee 6060-6130: Osservando queste linee, si notano due cose interessanti: l'uso dei cicli e l'attivazione di TI$ per la registrazione dell'ora nella memoria del C16.
Lo scopo del ciclo definito alla linea 6060 è di continuare a chiedere l'input dell'ora finché non viene battuta nel formato prestabilito. Il segno che conferma l'esattezza del formato è il valore della variabile RIGHT, cui viene assegnato il valore uno ad ogni inizio del ciclo, e che viene azzerata se un errore provoca un salto alle linee che iniziano da quella con il numero 6160. Il ciclo usa il valore di RIGHT come valore di una condizione alla linea 6060, più o meno come se fosse un'istruzione IF. Tutto ciò che, nella linea di programma, viene dopo IF è definito come condizione e il C16 normalmente traduce queste condizioni, come A>10, in due valori possibili: zero se la condizione è falsa e meno uno se è vera. L'istruzione IF (o in questo caso UNTIL) non viene eseguita nel caso in cui il valore della condizione sia zero, viene eseguita in tutti gli altri casi. Quindi, fintantoché il valore di RIGHT è zero (indicando così che è stato commesso un errore), l'istruzione UNTIL alla linea 6060 mantiene il ciclo in esecuzione, non appena cambia, smette di eseguirlo.
L'impostazione dell'ora sul C16 è la semplicità stessa. Quando il C16 viene acceso per la prima volta, il suo clock interno viene azzerato e comincia a contare in sessantesimi di secondo. Per vedere il risultato di questo conteggio, basta battere l'istruzione PRINT TI, dato che TI è una variabile scelta dallo stesso C16 per contenere l'ora. Usando l'istruzione, vi accorgerete immediatamente che la visualizzazione dell'ora in sessantesimi di secondo non ha molto senso. Per questo il C16 è dotato di un altro strumento di lettura, la variabile TI$, che ha un formato lungo sei cifre, due per l'ora, due per i minuti e due per i secondi. Così il valore 091245 va letto come un orologio che segna le ore 9, 12 minuti e 45 secondi.
Ma TI$ non è utile solo per la lettura dell'ora: il sistema è infatti predisposto in modo tale che, appena l'utente batte un nuovo valore per TI$, il C16 è in grado di regolare il suo orologio interno sull'ora specificata. Le linee dalla 6110 alla 6120 accettano una stringa di quattro caratteri che specifica le ore e i minuti e assegnano a TI$ questo valore con i secondi azzerati.

Come noterete, poiché è già stato predisposto un TRAP, non è più necessario verificare che l'ora battuta dall'utente sia esatta: ci pensa il C16 a farlo per voi.

Verifica

Battete:

GOTO 6000[RETURN]

e dovrebbe comparire la richiesta di introdurre l'ora in ore e minuti. La battitura di un formato non corretto provoca la comparsa di un messaggio di errore, mentre qualsiasi ora ragionevole non provocherà problemi di sorta. Dopo aver acquisito l'ora battuta, il programma si ferma con il messaggio di errore RETURN WITHOUT GOSUB. Niente male, non appena il modulo finale sarà stato introdotto, diventerà una subroutine di nome GOSUB.

Modulo 1.1.4: Impostazione del quadrante

Una volta battuta l'ora, possiamo passare a disegnare il quadrante dell'orologio su cui gli altri moduli apporranno le lancette.

Modulo 1.1.4: Linee 5000-5130

```
5000 REM*********************************
5010 REM DISEGNA IL QUADRANTE
5020 REM*********************************
5030 GRAPHIC1,1
5040 RESTORE5120
5050 FOR I=1 TO 12
5060 READ X,Y
5070 CHAR,X,Y,STR$(I)
5080 NEXT
5090 FOR I=90TO94
5100 CIRCLE ,160,100,I,I
5110 NEXT I
5120 DATA
24,4,27,7,28,12,27,17,24,20,18,22,13,20,10,17,
9,12,10,7,12,4,18,2
5130 RETURN
```

Commento

Linee 5040-5080 e 5120: Queste linee si limitano semplicemente a leggere le coordinate dei punti in cui dovranno essere posti i numeri da uno a 12 del quadrante. Le cifre dell'istruzione DATA sono a coppie, ciascuna costituita dalle coordinate X (orizzontale) e Y (verticale) di ogni ora. Le ore sono fornite dal ciclo che viene eseguito dodici volte.

Linee 5090-5110: Questo ciclo disegna quattro cerchi concentrici, il cui raggio è un pixel più lungo dell'ultimo. L'effetto è quello di creare intorno al quadrante un bordo più grosso di quello che si potrebbe ottenere con un solo cerchio. Noterete, al momento dell'esecuzione del programma, che il bordo non è perfettamente delineato e un po' di marone esce dai contorni. Ciò dipende da fatto che il C16 fa del suo meglio per disegnare un cerchio su una griglia di pixel disposti per linee rette e i suoi cerchi non possono quindi essere perfetti.

Verifica

Battete:

RUN 5000 [RETURN]

e dovreste vedere il C16 passare nel modo ad alta risoluzione e disegnare il quadrante. Terminato il disegno, lo schermo dovrà tornare di nuovo nel modo a bassa risoluzione visualizzando il messaggio RETURN WITHOUT GOSUB.

Modulo 1.1.5: Calcolo dei minuti e delle ore

In questo modulo, cominciamo ad entrare nel vivo del programma, che inizia con l'estrazione del valore dell'ora e dei minuti dalla variabile di sistema TI$.

Modulo 1.1.5: Linee 3000-3080

```
3000 REM*********************************
3010 REM CALCOLA I MINUTI
3020 REM*********************************
3030 DO UNTIL TT<>T0
3040 TT=VAL(LEFT$(TI$,4))
3050 LOOP
3060 H2=INT(TT/100)
3070 M2=TT-100*H2
3080 RETURN
```

Commento

Linee 3030-3050: Questo ciclo continua ad estrarre il valore delle prime quattro cifre della stringa TI$, cioè i caratteri che contengono le ore e i minuti. Il processo dura fino a quando TT, il valore dei minuti estratto, è diverso dal valore di TO, che è il valore di TT nel momento in cui è iniziata l'esecuzione. In altre parole, il programma rimane bloccato in questo ciclo finché i minuti non cambiano.

Linea 3060-3070: Queste linee esprimono una tecnica molto usata per trovare la cifra più alta e quella più bassa di un numero. Prendiamo, ad esempio, il numero 3472. Qual è il numero che sottratto da 3472 da' 72? La soluzione, se chiamiamo questo numero N, consiste nell'ottenere INT (N/100) o N/100 senza resto, e poi nel moltiplicarlo di nuovo per 100. In questo modo 3472/100=34.72 e l'intero è 34, che moltiplicato di nuovo per 100, ci dà il numero da sottrarre per ottenere le due cifre che formano le unità cercate. Con questa soluzione, dovreste essere in grado di capire che le linee ricavano il valore corrente per le ore e i minuti, e li pongono poi rispettivamente nelle variabili H2 e M2.

Verifica

Battete:

RUN3000[RETURN]

a quasi istantaneamente vedrete che l'esecuzione si ferma mentre compare il messaggio di errore RETURN WITHOUT GOSUB. Battete allora:

?TT[RETURN] — ? è la forma abbreviata di PRINT
?H2[RETURN]
?M2[RETURN]

Il valore di TT rappresenta le ore e i minuti, come risultato in TI$. H2 dovrebbe essere uguale alla parte corrispondente alle ore ed M2 a quella dei minuti.

Modulo 1.1.6: Il disegno delle lancette

Dopo aver calcolato tutte le cifre necessarie per l'ora, possiamo ora procedere ad usare il comando CIRCLE, uno strumento particolarmente flessibile del C16 che ci consentirà di disegnare le lancette dell'orologio. Se non avete chiaro l'uso di questo comando, vi consigliamo di leggere attentamente il manuale prima di procedere con la nostra analisi.

Modulo 1.1.6: Linee 4000-4110

```
4000 REM*********************************
4010 REM DISEGNA LE LANCETTE
4020 REM*********************************
4030 A2=INT(360*M2/60)
4040
CIRCLE0,160+FNCM(A1),100+FNSM(A1),3,40,0,360,A
1,120
4050
CIRCLE0,160+FNCH(A3),100+FNSH(A3),5,25,0,360,A
3,120
```

```
4060
CIRCLE,160+FNCM(A2),100+FNSM(A2),3,40,0,360,A2
,120
4070 A1=A2
4080 A2=INT(360*M2/12)+INT(M2/2)
4090
CIRCLE,160+FNCH(A2),100+FNSH(A2),5,25,0,360,A2
,120
4100 A3=A2
4110 RETURN
```

Commento

Essenzialmente, il modulo cancella la lancetta dell'ultimo minuto e la ridisegna in una nuova posizione, ripetendo poi l'operazione per la lancetta delle ore.

Linea 4030: A2 è l'angolo della lancetta dei minuti, con zero in corrispondenza delle 12.
Linea 4040: Questa linea cancella la lancetta corrente. Per sapere qualcosa di più sul significato delle parti che compongono la linea, ecco un breve commento:

CIRCLE0,160+FNCM(A1),100+FNSM(A1),3,40,0,360,A1,120

COMMENT 1 —-2—- —-3—- -4- -5- 6 7

1) L'origine del colore, con zero che specifica il colore di fondo (stiamo cancellando qualcosa).
2) La coordinata X del centro della figura da disegnare. 160 è la coordinata X del centro del quadrante dell'orologio. FNSM, come abbiamo visto, è una delle due funzioni necessarie a definire un punto sulla circonferenza di un cerchio. Ciò che intendiamo fare è trovare un punto sulla circonferenza all'interno del quadrante, con un raggio di 40 pixel in rapporto ai 90 per l'intero quadrante (sarà poco meno della metà della distanza fra il centro e il bordo del quadrante). L'angolo A1, che indica dove sulla circonferenza di questo piccolo cerchio sarà il centro della figura, è zero quando il programma inizia, ma è normalmente fissato sull'ultimo valore dei minuti.
3) Come per 2) ma relativamente alla coordinata Y.
4) La figura che stiamo disegnando avrà una larghezza di tre e un'altezza di 40 pixel.
5) Il disegno inizia da zero gradi e finisce a 360 (cioè la figura verrà disegnata con un bordo completo).
6) Oltre ad essere A1 gradi intorno al cerchio più piccolo, la figura viene fatta ruotare di A1 gradi. Quindi se la posizione sul cerchio più piccolo è su 15 minuti (90 gradi), anche la lancetta sarà su 90 gradi.

7) I punti usati per disegnare la figura saranno a 120 gradi l'uno dall'altro (disegnamo infatti un triangolo).
Detto tutto questo, è facile dedurre che ciò che disegneremo è un piccolo triangolo che punta verso il bordo esterno del quadrante che rappresenta i minuti, anche se nel caso di questa linea particolare, la figura viene disegnata solo per cancellare ciò che c'è già sullo schermo.

Linea 4050: Uguale alla linea precedente con la sola differenza che la lancetta cancellata è quella delle ore, che è leggermente più corta e il cui valore è conservato nella variabile A3.

Linea 4060: Esattamente uguale alla linea 3200 con la sola differenza che il colore usato è quello del disegno e che l'angolo A2 è il nuovo valore per i minuti. Questa linea ridisegna la lancetta dei minuti in una nuova posizione.

Linee 4070-4080: Il valore corrente per i minuti è registrato in A1 in modo che la prossima volta possa essere cancellato, poi viene usata A2 per memorizzare l'angolo della lancetta delle ore. Essa sarà di 30 gradi per ogni ora esatta e di un grado ogni due minuti.

Linee 4090-4100: La lancetta delle ore viene ridisegnata e le ore memorizzate in A3 in modo da poterle poi cancellare.

Verifica

Per predisporre il modulo alla verifica, occorre introdurre un certo numero di linee provvisorie. Poiché il programma completo ha bisogno dell'aggiunta di solo poche linee, è consigliabile lasciare la verifica a dopo il completamento del programma.

Modulo 1.1.7: Come far funzionare il tutto

A questo punto vi starete chiedendo perché il programma è scritto in questo modo. Non sarebbe stato meglio scrivere le funzioni insieme e farle eseguire da istruzioni GOTO? Purtroppo questo è il modo in cui è scritta la maggior parte dei programmi pubblicati. In questo libro invece tutti i programmai sono costituiti da moduli ben identificabili. La ragione di questa scelta è che i programmi scritti in moduli sono più facili da leggere, possono essere sottoposti a debug in modo più semplice, possono essere perfezionati mediante l'aggiunta di nuovi moduli nel caso in cui si apprendano tecniche più efficaci e si prestano più facilmente ad espansioni mediante la semplice aggiunta di moduli nuovi, come vedremo nell'ultimo programma di questo capitolo. Ci sono molte cose che si possono imparare analizzando i programmi contenuti in questo libro, ma certo quella più importante per la vostra abilità di programmatori è la tecnica della programmazione modulare.
Il modulo che stiamo analizzando ora è la chiave di questa tecnica, perché nel momento in cui tutti i moduli funzionanti sono stati introdotti e verificati, si ha sempre bisogno di un modulo particolare con cui poter controllare il flusso completo del programma. In un certo senso, il modulo che state per introdurre è il programma vero e proprio, tutto il resto non è altro che un suo ampliamento.

Modulo 1.1.7: Linee 2000-2120

```
2000 REM*********************************
2010 REM UNITA' DI CONTROLLO
2020 REM*********************************
2030 GOSUB 6000
2040 GOSUB 5000
2050 TRAP 2110
2060 DO
2070 GOSUB 3000
2080 GOSUB 4000
2090 T0=TT
2100 LOOP
2110 GRAPHIC0
2120 END
```

Commento

Linee 2030-40: Viene introdotta l'ora e viene disegnato il quadrante.

Linee 2050,2110-20: Questo è quanto permette di interrompere il programma. Premendo STOP, che genera un BREAK ERROR, si ottiene il ritorno del C16 al modo a bassa risoluzione e la fine (END) dell'esecuzione. Ancora una volta, è meglio non introdurre il comando TRAP finché non si è certi che il resto del programma funzioni correttamente, dato che impedirebbe la generazione di messaggi di errore in caso di malfunzionamento del programma.

Linee 2060–2100: Il ciclo del programma principale, che richiama il modulo per il calcolo dei minuti e quello per il disegno delle lancette. La variabile TO, usata alla linea 2000 del modulo, serve a decidere se il valore dei minuti è cambiato.

Verifica

Il programma dovrebbe ora funzionare perfettamente. Eseguitelo (RUN), battete l'ora e vedrete comparire il quadrante con le lancette al posto giusto.

Conclusioni

Questo programma presenta una caratteristica osservabile solo quando è stato interamente introdotto. Se lo osservate, noterete che non contiene neppure un'istruzione GOTO. È un fatto che vi colpirà certamente, soprattutto se avete imparato i primi rudimenti della programmazione su qualche macchina meno potente del C16. La cosa è stata voluta, non si tratta di un caso. Negli ultimi anni il BASIC si è sviluppato nel senso di una costante eliminazione delle istruzioni GOTO.

Anche se questa scelta può portare a programmi molto lunghi e ad un grande

spreco di memoria, esistono delle ottime ragioni per cercare di ridurre al minimo l'uso del GOTO nei programmi. L'istruzione è infatti troppo arbitraria: è l'istruzione ad eseguire un salto che però avrà effetto solo al momento della lettura del programma, cioè in un momento successivo. Prendete ad esempio il ciclo alle linee 6060 e 6130. Esso avrebbe potuto essere eseguito in questo modo:

6130 IF RIGHT=0 THEN 6060 (GOTO è implicato dopo il THEN)

Se un'istruzione di questo tipo fosse eseguita, il flusso del programma subirebbe un'interruzione. Nella sua forma attuale, essa annuncia esattamente che cosa farà (fa' qualcosa finché non succede qualcos'altro). È chiaro come concetto e sarà chiaro quando il programma verrà letto in un momento successivo. Cominciate ad usare l'istruzione GOTO e il programma si trasformerà brevemente in un numero incontrollato di salti difficili da pianificare o da spiegare e, cosa ancora peggiore, sarete incoraggiati a mettere insieme un coacervo di istruzioni incomprensibili che farete più presto a riscrivere che non a risistemare. Lo scopo della programmazione con una macchina che mette a disposizione il ciclo DO è quello di scrivere programmi che fluiscano dall'inizio alla fine senza salti arbitrari. Non ha senso impazzire sui programmi (fra i programmatori si dice che un GOTO in un programma è già il segno di una mente contorta!), ma soprattutto è bello scoprire quanta soddisfazione può dare sviluppare ed eseguire un programma in grado di lavorare in base alle cose che succedono finché o mentre, e non saltando invece qua e là senza alcun controllo diretto da parte del programmatore.

Programma 1.2: Orologio II

Scopi del programma

Una delle cose più piacevoli dei computer con immagini grafiche di buona qualità, come quelle del C16, è che esse permettono di divertirsi riproducendo oggetti in modi nuovi e fantasiosi. Dopo aver imparato a creare un orologio normale, con il programma che segue giocheremo a riprodurre le ore in un modo completamente diverso. Qui le ore e i minuti sono rappresentati da due linee che scorrono da sinistra a destra e dall'alto al basso, dividendo lo schermo in quattro rettangoli di diverso colore. Molte delle istruzioni del primo programma sono ripetute nel secondo, per cui alcune spiegazioni risulteranno abbreviate.
Gli argomenti completamente nuovi sono in pratica due:
1) la suddivisione della stringa
2) la stampa verticale di stringhe

Modulo 1.2.1: Impostazione dell'ora

Questo primo modulo equivale a quello corrispondente del programma Orologio analogico. La sola differenza fra i due sta' nel fatto che in questo le ore e i minuti vengono introdotti separatamente. La verifica invece è uguale a quella eseguita nel primo programma.

Modulo 1.2.1: Linee 2000-2130

```
2000 REM*********************************
2010 REM INSERIMENTO ORE
2020 REM*********************************
2030 TRAP 2110
2040 DO UNTIL RIGHT : RIGHT=1
2050 SCNCLR
2060 INPUT "[CLEAR][GRN][CD]INSERISCI LE ORE
(01 - 12):";H$
2070 INPUT "[CD][RED]INSERISCI I MINUTI (00 -
59):";M$
2080 TI$=H$+M$+"00"
2090 LOOP
2100 TRAP : RETURN
2110 PRINT "[CD][CD]INSERIMENTO ERRATO -
RIPROVA" : RIGHT=0
2120 FOR I=1 TO 3000 : NEXT
2130 RESUME NEXT
```

Modulo 1.2.2: Impostazione della griglia

Il modulo stampa una griglia delle ore e dei minuti lungo il lato sinistro e quello superiore dello schermo.

Modulo 1.2.2: Linee 3000–3050

```
3000 REM*********************************
3010 REM VIDEO
3020 REM*********************************
3030 PRINT "[CLEAR][WHT]    5[RVS ON] 10[RVS
OFF] 15[RVS ON] 20[RVS OFF] 25[RVS ON] 30[RVS
OFF] 35[RVS ON] 40[RVS OFF] 45[RVS ON] 50[RVS
OFF] 55[RVS ON] 60[RVS OFF]  "
3040 FOR I=1 TO 12 : CHAR
,0,I*2,MID$(STR$(I),2) : NEXT
3050 RETURN
```

Commento

Linea 3030: I valori relativi ai minuti spaziati lungo il bordo superiore dello schermo in bianco.

Linea 3040: Questa linea stampa le ore dall'alto verso il basso lungo il lato sinistro dello schermo, utilizzando il valore della variabile I del ciclo. L'espressione

MID$ è in quel punto semplicemente per togliere lo spazio aggiunto all'inizio di un numero positivo al posto del segno +.

Verifica

Per verificare la correttezza del modulo, inserite temporaneamente la linea:

```
3045 GETKEY A$
```

Scopo di questa istruzione è di evitare che lo schermo scorra verso l'alto con il messaggio RETURN WITHOUT GOSUB. Ora battete:

```
GOTO 3000 [RETURN]
```

e intorno al bordo dello schermo vedrete comparire una griglia. Quando siete soddisfatti, premete STOP e togliete la linea provvisoria.

Modulo 1.2.3: Calcolo delle ore e dei minuti

Questo modulo corrisponde al modulo 1.1.5 del programma precedente, dal quale si discosta solo per il fatto di contenere alcune istruzioni per la riproduzione delle ore e dei minuti in base alle dimensioni dello schermo.

Modulo 1.2.3: Linee 4000-4100

```
4000 REM*********************************
4010 REM CALCOLA LE ORE
4020 REM*********************************
4030 TT=VAL(LEFT$(TI$,4))
4040 H=INT(TT/100)
4050 M=TT-100*H
4060 H=H*2 : M=INT(M*0.6+0.5)
4070 IF M>=18 THEN H=H+1
4080 IF H>=24 THEN H=H-24
4090 IF M=0 THEN M=1
4100 RETURN
```

Commento

Linee 4030-4050: Come in orologio analogico.

Linee 4060-4090: Nel visualizzare il trascorrere del tempo, avremo a nostra disposizione 24 linee per le ore e 36 colonne per i minuti. Dovremo quindi fare qualche modifica ai valori di ore e minuti per poterli adattare allo schermo. Inoltre, poiché ci sono due linee per ogni ora, l'ora può essere spostata di mezza posizione ogni volta che il valore dei minuti è trenta. Un'ultima modifica è quella necessaria per fare in modo che non sia mai l'ora esatta dato che il video non risulterebbe chiaro.

Verifica

Battete:

GOTO 4000[RETURN]

e dovreste veder comparire il messaggio RETURN WITHOUT GOSUB. Generate TT, H ed M. (Se TT risulta maggiore di 1200, cioè dopo le 12 nel sistema a 24 ore, riprovate battendo ?TT-1200). Dovreste scoprire che H è uguale a due volte il valore delle prime due cifre di TT (o due volte più uno se i minuti indicano che è trascorsa la mezza) e M è uguale a circa i due terzi del valore delle ultime due cifre.

Modulo 1.2.4: Visualizzazione dell'ora

In questo modulo arriviamo finalmente al problema di visualizzare sullo schermo i rettangoli che ci permettono di riprodurre il trascorrere delle ore. Ciò che il modulo deve ottenere è l'effetto di una linea che scorre trasversalmente allo schermo per indicare i minuti e di un'altra che scende dal bordo superiore verso quello inferiore indicando le ore. L'effetto è ottenuto dividendo lo schermo in quattro settori rettangolari nei colori porpora, rosso, verde e bianco e ricavando visivamente le linee con i loro bordi.

Figura 2

RETTANGOLO 1 (PORPORA)	LINEA DEI MINUTI	RETTANGOLO 2 (ROSSO)
RETTANGOLO 3 (ROSSO)		RETTANGOLO 4 (BIANCO)

Indubbiamente si potrebbe porre ciascun rettangolo direttamente sullo schermo nel posto giusto, ma questo rappresenta una difficoltà inutile. Tutto quello che ci serve è tracciare trasversalmente righe di spazi colorati che cambiano opportunamente dal porpora al rosso nella metà superiore dello schermo e dal verde al bianco in quella inferiore. Le linee vengono create utilizzando la tecnica della suddivisione della stringa.

Module 1.2.4: Linee 5000-5140

```
5000 REM*******************************
5010 REM VISUALIZZA LE ORE
```

```
5020 REM*********************************
5030 M1$="[CR][CR][PUR][RVS ON]
                          " : REM 36 SPAZI
5040
M1$=LEFT$(M1$,M+4)+"[RED]"+RIGHT$(M1$,36-M)
5050 M2$="[CR][CR][GRN][RVS ON]
                          " : REM ANCORA 36 SPAZI
5060
M2$=LEFT$(M2$,M+4)+"[WHT]"+RIGHT$(M2$,36-M)
5070 PRINT "[HOME][CD]";
5080 IF H>0 THEN FOR I=1 TO H:PRINT M1$: NEXT
5090 IF H<23 THEN FOR I=H+1 TO 23:PRINT M2$:
NEXT
5100 PRINT M2$;
5110 CHAR ,38,7,"[RVS OFF][WHT]"
5120 DT$=LEFT$(TI$,2)+"  "+MID$(TI$,3,2)+"
"+RIGHT$(TI$,2)
5130 FOR I=1 TO LEN(DT$):PRINT
MID$(DT$,I,1);"[CD][CL]";:NEXT
5140 RETURN
```

Commento

Linea 5030: La variabile di stringa M1$ viene posta uguale a due caratteri di controllo del cursore, quello del porpora e il carattere RVS ON, seguiti da 36 spazi. La stampa di questa stringa all'inizio di una linea da' come risultato una stringa di 36 spazi color porpora, rientrati di due posizioni di carattere rispetto al margine sinistro dello schermo.

Linea 5040: A questo punto, impiegando la tecnica della suddivisione della stringa, estraiamo dalla parte sinistra della stringa tutto quanto serve per trasportare la linea di spazi fino alla linea dei minuti, per aggiungere alla fine di quella parte di stringa il carattere di controllo del rosso e poi il resto della stringa originale. La variabile M è già stata predisposta per rappresentare il numero di minuti su una scala di 1-36 invece di 1-60, quindi non sono necessari aggiustamenti. Poiché la stringa di spazi è preceduta da quattro caratteri di controllo, abbiamo bisogno di prendere i primi M+1 caratteri per la larghezza del rettangolo (RETTANGOLO 1) nella Figura 2.
Dopo aver aggiunto il carattere di controllo per far diventare gli spazi rossi, e dopo aver ricordato che la larghezza del video è di 36 spazi, tutto ciò che rimane da fare è aggiungere gli ultimi 36-M caratteri della stringa originale. Questa possibilità di estrarre parti di stringa sulla base di un preciso calcolo è un aspetto molto interessante e utile del linguaggio BASIC ed è molto importante che ne

abbiate capito il meccanismo fin da questo esempio.

Linee 5050-5060: La stessa procedura viene applicata a M2$ per creare una linea verde e bianca.

Linea 5070: Il cursore viene rimandato nell'angolo in alto a sinistra dello schermo e poi in giù di una posizione in modo che i valori dei minuti in alto non vengano cancellati.

Linee 5080-5100: Questi due cicli riproducono abbastanza della linea porpora/rosso da portare il lato inferiore dei rettangoli uno e due nella **Figura 2** sulla linea delle ore, e poi riempiono il resto del video con i rettangoli tre o quattro. L'ultima linea del video viene riprodotta sullo schermo con un punto e virgola in modo da evitare lo scorrimento dell'immagine.

Linee 5110-5130: Ecco un'altra occasione in cui viene usata la tecnica della suddivisione della stringa. Questa volta, però, essa serve a riprodurre l'ora in forma digitale lungo il lato destro dello schermo. L'effetto viene ottenuto prendendo la variabile di sistema TI$, dividendola in tre tronconi, che rappresentano le ore, i minuti e i secondi, e separandoli tra loro con l'inserimento di spazi. Viene quindi riprodotta sullo schermo, carattere dopo carattere, la stringa DT$ (Digital Time$) che ne risulta, partendo dal punto 38,7 dello schermo; la posizione e il colore (bianco) vengono fissati con CHAR. Non appena sono stati riprodotti i caratteri "cursore giù" e "cursore a destra" che permettono di portare la posizione di stampa (spesso si usa il termine stampa anche parlando del video) sullo spazio-carattere immediatamente sotto l'ultimo carattere riprodotto. In altre parole ciò che viene stampato è esattamente questo:

```
CARATTERE   !
            !
            !
            V
<---------
```

Verifica

Battete, nel modo diretto:

```
H=12 [RETURN]
M=18[RETURN]
GOTO 5000
```

A questo punto dovreste vedere lo schermo suddiviso in quattro rettangoli abbastanza uguali lungo le linee della figura qui in alto. Il programma si fermerà poi con il messaggio di errore RETURN WITHOUT GOSUB.

Modulo 1.2.5: Assemblaggio dei moduli

Dopo aver introdotto tutti gli elementi del programma, perfettamente funzionanti,

possiamo passare a costruire un modulo di controllo per poterli eseguire nel modo corretto.

Modulo 1.2.5: Linee 1000-1120

```
1000 REM*********************************
1010 REM CICLO DI CONTROLLO
1020 REM*********************************
1030 COLOR 4,1 : COLOR 0,1
1040 GOSUB 2000
1050 GOSUB 3000
1060 TRAP 1110
1070 DO
1080 GOSUB 4000
1090 GOSUB 5000
1100 LOOP
1110 SCNCLR
1120 END
```

Commento

Linee 1030-1050: Lo schermo e il bordo esterno vengono fatti diventare neri e vengono aggiunti l'ora e la griglia esterna.

Linee 1060 e 1110-1120: Poiché il programma è corretto è possibile commettere qualsiasi errore perché in questo caso il programma salterebbe alle linee che azzerano lo schermo e porrebbe fine al programma. Quest'ultimo è quindi interrotto premendo il tasto RUN/STOP.

Linee 1070-1100: Finché non viene premuto il tasto RUN/STOP, il programma continua a calcolare le variabi H ed M e a visualizzare l'ora.

Verifica

Siete ora in grado di eseguire l'intero programma, battere l'ora e vederla riprodotta nel modo desiderato.

Programma 1.3: Timer

Scopi del programma

Abbiamo già notato nel primo programma che uno dei punti di forza della programmazione modulare è la possibilità che essa offre di fare aggiunte al programma senza l'insorgere di inconvenienti nelle sue parti già esistenti. Per dimostrarlo daremo un'occhiata ad un altro programma sulla misurazione del tempo, che è sempre utile avere a propria disposizione, e con il prossimo introdurremo nuove funzioni per avere così un quadro completo di tutte le potenzialità del C16 in questo campo.

Il programma, così come si presenta, permette di ottenere 10 timer flessibili con conto alla rovescia, ciascuno programmabile a parte in modo da generare, dopo un certo intervallo di tempo, un segnale acustico e un messaggio che ne spiega i motivi.
L'ultimo programma di questo primo capitolo non è altro che un potenziamento di questo. Esso è infatti in grado di tenere 16 appuntamenti quotidiani per un periodo di sette giorni, visualizzandoli a comando sullo schermo e continuando a controllare i dieci timer per vedere se i segnalatori devono suonare.
Per poter introdurre la seconda versione del programma, è indispensabile introdurre prima questa.
Osservando il programma, noterete che la sua forma è diversa da quella dei programmi finora analizzati. Siccome è il più lungo e il più complesso, per aiutarvi ad individuare le unità che lo costituiscono abbiamo inserito delle linee vuote. Naturalmente queste linee non sono necessarie al funzionamento del programma possono, volendo, essere tralasciate. Di queste linee non si è fatto molto uso nel corso del programma per la semplice ragione che lo spazio in più necessario avrebbe allungato troppo questo libro che volevamo maneggevole.

Figura 1.2 Timer

```
TIMER E IMPEGNI

sono le ore:
 0  - 16-23-34 - TELEFONARE PIERO
 1  - 16-44-05 - PARTITA IN TV
 2  - 18-37-08 - ANDARE DA MARIA
 3  - 00-00-00 -
 4  - 00-00-00 -
 5  - 00-00-00 -
 6  - 00-00-00 -
 7  - 00-00-00 -
 8  - 00-00-00 -
 9  - 00-00-00 -
```

Il programma ci permette di introdurre tre cose nuove:
1) il modulo menu
2) l'uso di SOUND
3) l'uso di titoli molto semplici

Modulo 1.3.1: Un titolo facile facile

Avere un titolo di presentazione in alto sullo schermo nei vari momenti dell'esecuzione del programma può cambiare spesso l'atteggiamento dell'utente. Questo piccolo modulo rappresenta uno dei tanti modi possibili per raggiungere lo scopo. Può essere chiamato in qualsiasi momento e azzera lo schermo e vi colloca in alto il nome del programma.

Modulo 1.3.1: Linee 14000-14120

```
14000 REM*******************************
14010 REM TITOLO
14020 REM*******************************
14030 :
14040 :
14050 COLOR0,1 : COLOR 1,9,5 :SCNCLR
14060 CHAR ,12,0,"┌──────────┐"
14070 CHAR ,12,1,"│   TIMER  │"
14080 CHAR ,12,2,"└──────────┘"
14090 PRINT
14100 RETURN
14110 :
14120 :
```

Commento

Linee 14060-14080: I caratteri grafici sono sui tasti U, C, I, B J e K.

Verifica

Eseguite (RUN) il modulo e vedrete comparire la parola TIMER in un riquadro posto in alto sullo schermo.

Modulo 1.3.2: Formato cronologico

Linee come queste, le abbiamo già incontrate nel primo programma di questo capitolo. La loro funzione consiste nel prendere il valore della variabile di sistema TI (non TI$), che registra il tempo in sessantesimi di secondo dal momento in cui il C16 viene acceso o da quando viene azzerato il suo orologio, e nel ricavare una stringa nella forma 00-MM-SS (ore, minuti, secondi) da usare nel programma.

Modulo 1.3.2: Linee 7000-7170

```
7000 REM*******************************
7010 REM STRINGHE CON IL TEMPO
7020 REM*******************************
7030 :
7040 :
```

```
7050 TT=INT(TT/60)
7060 HH=INT(TT/3600)
7070 MM=INT(TT/60)-60*HH
7080 SS=TT-60*MM-3600*HH
7090 :
7100 :
7110 HH$ = RIGHT$("00"+MID$(STR$(HH),2),2)
7120 MM$ = RIGHT$("00"+MID$(STR$(MM),2),2)
7130 SS$ = RIGHT$("00"+MID$(STR$(SS),2),2)
7140 TT$=HH$+"-"+MM$+"-"+SS$
7150 RETURN
7160 :
7170 :
```

Commento

Linee 7050-7080: La variabile TT viene inviata a questa subroutine uguale alla variabile di sistema TI, di cui abbiamo già parlato. TT viene immediatamente divisa per 60 in modo da diventare l'ora espressa in secondi. Da quest'ultima vengono poi ricavate le ore, i minuti e i secondi.

Linee 7110-7140: Le variabili di stringa servono a ricavare le ore, i minuti e i secondi. L'operazione viene eseguita in modo tale da essere sicuri che, se un numero ha meno di due cifre, viene completato da uno zero. Ancora una volta notate che, quando si estrae una stringa da un numero usando STR$, la parte utile della stringa comincia dal carattere due, dato che, davanti ai numeri positivi, viene aggiunto uno spazio in modo da avere una corrispondenza con la posizione del segno meno nei numeri negativi.

Verifica

Battete:

```
TT = TI [RETURN]
GOTO 7000
```

e scoprirete subito che l'esecuzione ha termine con la comparsa del messaggio RETURN WITHOUT GOSUB. Ora battete:

```
?TI$ [RETURN]
?TT$[RETURN]
```

TI$ rappresenta il tempo pochi secondi dopo TT$, sempreché siate stati abbastanza rapidi, ma offre il vantaggio di fornire il tempo nella forma più chiara, cioè in ore, minuti e secondi, nettamente separati gli uni dagli altri.

Modulo 1.3.3: Messaggi di errore

A differenza dei due programmi che lo hanno preceduto, questo avrà poi la possibilità di generare diversi messaggi di errore. Per renderlo più chiaro, essi saranno memorizzati in un unico modulo, il quale, fino a quando non verrà aggiunto il prossimo programma, resta però costituito da un solo messaggio relativo al formato delle ore.

Modulo 1.3.3: Linee 11000-11050

```
11000 REM*********************************
11010 REM MESSAGGIO D'ERRORE
11020 REM*********************************
11030 PRINT "ORA INSERITA NON CORRETTAMENTE" :
RESUME
11040 :
11050 :
```

Modulo 1.3.4: Inizializzazione

Questo è un modulo molto semplice e linerare che permette di predisporre le poche variabili necessarie e di leggere l'ora battuta dall'utente.

Modulo 1.3.4:linee 1000-1200

```
1000 REM*********************************
1010 REM INIZIALIZZAZIONE
1020 REM*********************************
1030 :
1040 :
1050 GOSUB 14000 : COLOR 1,5,4 : PRINT
CHR$(142)
1070 DIM TT$(3),TT(3)
1080 :
1090 :
1100 Q$="" : DO UNTIL Q$="S"
1110 TRAP 11030
1120 INPUT "[CD][CD]INSERISCI LE ORE
(HHMMSS):";TI$
1130 TRAP
1140 TT=TI
1150 GOSUB 7000
1160 PRINT "[CD]SONO LE: ";TT$
1170 INPUT "[CD]E' ESATTO (S/N):";Q$
```

```
1180 LOOP
1190 :
1200 :
```

Commento

Linea 1050: Stampando CHR$(142), un carattere di controllo, si ottiene un programma tutto a lettere maiuscole. La maggior parte dei programmi di questo libro sono studiati per essere eseguiti in questo modo, anche se non necessariamente fissano il modo di funzionamento automaticamente. Il passaggio dalle maiuscole alle minuscole e viceversa può essere ottenuto premendo il tasto SHIFT e il logotipo COMMODORE contemporaneamente.

Linee 1100-1180: Le linee contengono una routine di impostazione delle ore del tutto normale, se si esclude il fatto che, dopo l'impostazione dell'ora, all'utente viene richiesta una conferma. Questo è un modo molto usato per avere la certezza che le ore siano esattamente quelle (se l'utente non conferma con un Y, il sistema richiede la ribattitura del dato.)

Verifica

Inserire temporaneamente la linea:

```
1999 END
```

ed eseguite (RUN) il programma. Per prima cosa comparirà il titolo del programma e potrete introdurre i dati corrispondenti all'ora, sicuri della loro correttezza in quanto il programma contiene un controllo del formato. A questo punto, dopo aver chiesto di confermare i dati appena battuti, il programma si fermerà per effetto dell'istruzione END, inserita temporaneamente e che dovrete togliere.

Modulo 1.3.5: Verifica dei timer

Questo piccolo modulo è parte integrante del programma, nel senso che gli consente di effettuare una sosta durante la quale l'utente può introdurre i dati, pur continuando a campionare i 10 timer per vedere se è giunto il momento di attivare qualche segnale acustico. Introdurremo questo modulo esattamente a questo punto, anche se esso non verrà usato se non quando non saranno stati introdotti altri moduli: esso è infatti una subroutine essenziale per il modulo menu di cui vi parleremo fra poco.

Modulo 1.3.5: Linee 15000-15090

```
15000 REM*******************************
15010 REM ATTESA DEL TIMER
15020 REM*******************************
15030 :
15040 :
```

```
15050 PRINT "[FLASH ON]SONO IN ATTESA"
15060 T$="" : DO UNTIL T$<>"" OR TM
15070 GOSUB 4000
15080 GET T$ : LOOP : TM=0
15090 RETURN
```

Commento

Linea 15050: Le parole SONO IN ATTESA riprodotte sullo schermo a lettere lampeggianti verranno usate per tutto il programma come indicazione del fatto che l'utente può introdurre certi dati. La natura esatta di questi dati varia in base al punto del programma dal quale il modulo viene invocato. Notate che quando vedete le parole SONO IN ATTESA non dovete premere RETURN dopo i dati perché il programma agisce istantaneamente su qualsiasi tasto voi premiate.

Linee 15060-15080: Questo ciclo continua a chiamare la subroutine alla linea 4000 (che verrà introdotta più avanti) e le fa' seguire la verifica con GET per vedere se è stato premuto un tasto. Se la chiamata a 4000 provoca la generazione di un segnale acustico o se viene premuto un tasto, il modulo ha terminato e il programma ritorna al punto in cui è avvenuta la chiamata.

Verifica

Per eseguire la verifica del modulo, inserite provvisoriamente la linea:

```
4000 RETURN
```

e poi la linea:

```
GOTO15000[RETURN]
```

e vedrete comparire sullo schermo le parole SONO IN ATTESA a lettere lampeggianti. A questo punto non succederà più niente finché non premete un tasto, che sarà poi seguito dal messaggio di errore RETURN WITHOUT GOSUB. Attenzione: per il momento non togliete la linea 4000 provvisoria.

Modulo 1.3.6: Il menu del programma

Arriviamo così ad una delle novità di cui parlavamo all'inizio e che giocherà un ruolo importante nei programmi riportati in questo libro, vale a dire il menu. Nei programmi che ci hanno permesso di giungere a questo, il controllo era lasciato al programma stesso. Una volta lanciato il comando RUN, un apposito modulo assumeva il controllo del flusso di esecuzione del programma fino a quando l'utente non esprimeva la volontà di porvi fine. Questo programma, invece, come molti altri che seguiranno, è diverso, nel senso che non esiste un'unica direzione da seguire. Esso presenta cioè all'utente una serie di possibilità e deve essere quest'ultimo a dettare ciò che deve succedere. Questo viene realizzato per mezzo di un modulo noto come "menu del programma", menu perché presenta all'utente un elenco di scelte che il programma può offrire. Altri programmi più complessi,

che incontreremo più avanti, faranno uso di diversi menu, tutti selezionabili all'interno di un menu principale.
Per il momento comunque ci limiteremo a vederne uno solo.

Modulo 1.3.6: Linee 2000-2280

```
2000 REM*********************************
2010 REM MENU
2020 REM*********************************
2030 :
2040 DO UNTIL Z=4
2050 GOSUB 14000
2060 COLOR 1,12,4
2070 PRINT "[CD][CD][CD]COMANDI DISPONIBILI:"
2080 COLOR 1,4,4
2090 PRINT "[CD][CD]   1) MODIFICA IL TIMER"
2100 PRINT "   2) VISUALIZZA IL TIMER"
2110 PRINT "   3) SCHERMO VUOTO ED ATTESA"
2150 PRINT "   4) STOP"
2160 COLOR 1,12,4
2170 PRINT "[CD][CD]LA TUA SCELTA: ";
2180 :
2190 :
2200 GOSUB 15000
2210 Z=VAL(T$)
2220 ON Z GOSUB 6000,5000,13000
2230 LOOP
2240 :
2250 :
2260 SCNCLR : END
2270 :
2280 :
```

Commento

Linee 2040 e 2230-2260: Questo ciclo continua a visualizzare il menu ogni volta che l'esecuzione ritorna a questo modulo, finché l'utente non batte il numero 7, che provoca la fine del programma.

Linee 2060-2170: La lista delle scelte offerte dal programma, assieme ai comandi di controllo del colore, che consentono di dare all'immagine sullo schermo un aspetto gradevole.

Linea 2200: Normalmente a questo punto del modulo menu c'è una normale istruzione INPUT. Il motivo per cui viene inserita questa linea di chiamata del modulo precedente è che vogliamo controllare continuamente lo stato dei timer, anche con il menu sullo schermo. L'effetto di questa chiamata della subroutine di campionamento è che T$ viene fornita completa dei dati battuti dall'utente.

Linee 2210-2220: L'input dell'utente viene trasferito nella variabile Z, poi il comando ON...GOSUB sceglie la destinazione dalla lista successiva corrispondente al numero battuto sulla tastiera. Se viene battuto un numero al di fuori di quelli della lista delle destinazioni di GOSUB, cioè se il numero è uguale a zero o è maggiore della lunghezza della lista, il comando non viene eseguito e il ciclo di istruzioni riporta di nuovo il menu sullo schermo.

Verifica

Avviate l'esecuzione. Il programma chiede che venga battuta l'ora, come con il modulo 1.3.3, e visualizza il menu. Battete un numero non valido per avere la conferma che non viene accettato. Scegliendo le opzioni da uno a tre si otterrà l'interruzione del programma con la comparsa del messaggio UNDEFINED LINE NUMBER, mentre battendo quattro l'interruzione del programma sarà accompagnata dall'azzeramento dello schermo.

Modulo 1.3.7: Visualizzazione dei valori correnti

Scopo di questo modulo è riprodurre in modo ordinato le ore (i minuti e i secondi) su cui sono regolati al momento i dieci timer. Poiché non abbiamo introdotto il modulo che ci permette di impostare i valori, i timer saranno ancora regolati su zero.

Modulo 1.3.7: Linee 5000-5360

```
5000 REM*********************************
5010 REM VISUALIZZA IL TIMER
5020 REM*********************************
5030 :
5040 :
5050 T$="" : DO UNTIL T$<>""
5060 :
5070 :
5080 GOSUB 14000: COLOR 1,6,4
5090 CHAR ,11,4,"[RVS ON]ORARI E IMPEGNI" :
PRINT
5100 PRINT "[CD]SONO LE ORE: [CD][CD]"
5110 FOR I=0 TO 9
```

```
5120 COLOR 1,4-(I AND 1),4
5130 PRINT I;" - ";
5140 TT=TT(I)
5150 GOSUB 7000
5160 PRINT TT$;" - ";TT$(I)
5170 NEXT I
5180 :
5190 :
5200 COLOR 1,6,4 : IF PART THEN PART=0 :
RETURN
5210 REM PART INDICA QUEL MODULO CHIAMATO AD
INTERROMPERE LA ROUTINE
5220 :
5230 :
5240 CHAR ,0,23,"[FLASH ON]ATTENDI"
5250 T$="" : DO UNTIL T$<>"" OR TM
5260 TT=TI
5270 GOSUB 7000
5280 CHAR ,14,6,TT$ : PRINT
5290 GOSUB 4000
5300 GET T$ : LOOP : TM=0
5310 :
5320 :
5330 LOOP
5340 RETURN
5350 :
5360 :
```

Commento

Linee 5050 e 5330: La visualizzazione continua fino a quando non viene premuto un tasto.

Linee 5110-5170: Quelli che abbiamo introdotto sono i due vettori contenenti le ore e i messaggi di ciascun timer. Osservate l'uso di una condizione logica per avere righe di testo a due colori alterni. L'espressione AND fra parentesi fà da spola fra il valore zero e quello che dipende dal fatto che il valore della variabile I del ciclo sia pari o dispari. Osservate inoltre l'uso di TT, TT0, TT$ e TT$0 (il C16 non incontra nessuna difficoltà nel distinguere fra variabili e vettori che hanno lo stesso nome).

Linea 5200: Il valore della variabile PART è determinato da un altro modulo che non abbiamo ancora esaminato. La funzione di questa linea è di riportare

l'esecuzione al modulo dopo aver riprodotto i valori dei timer. In altre parole, la linea rappresenta un modo per usare solo la parte del modulo corrente che è necessaria ad un'altra parte del programma.

Linee 5240-5300: Per diversi aspetti, una replica delle operazioni del modulo 1.3.4 che abbiamo già introdotto. Il motivo per cui è necessario introdurre una routine speciale è che, oltre all'attesa della battitura di un dato e al campionamento dei timer, il modulo ha bisogno che l'ora venga riprodotta sullo schermo e sia continuamente aggiornata. Poiché il modulo 1.3.4 non prevede questa possibilità, occorre scrivere una routine parallela.

Verifica

Battete:

RUN 5000[RETURN]

e vedrete comparire le ore dei dieci timer, che comunque sono rappresentate tutte dai valori 00-00-00 e non sono accompagnate dai messaggi, dato che non sono stati introdotti. In fondo allo schermo comparirà anche la parola ATTENDI lampeggiante e, sotto il titolo del programma, l'ora.

Modulo 1.3.8: Estrazione del valore di una stringa corrispondente all'ora

Abbiamo già avuto modo di osservare che il tempo che trascorre viene registrato dal C16 mediante due variabili di sistema, TI$ e TI, quest'ultima con il numero di sessantesimi di secondo trascorsi dal momento dell'accensione del computer. In diverse occasioni, nel corso del programma, useremo TI per ottenere l'ora corrente da confrontare con le ore su cui sono regolati i timer, anch'esse memorizzate sotto forma di un numero di sessantesimi di secondo. Quando l'utente batte i valori corrispondenti alle ore impostate per i timer, però, questi vengono introdotti sotto forma di stringa. Bene, lo scopo di questo modulo è quello di tradurre questa stringa in un numero. Se provate a confrontare questo modulo con il modulo 1.3.2, che avete già introdotto, vedrete che esso non è altro che l'immagine speculare di quello, che usa il metodo della suddivisione della stringa per isolare le diverse parti della stringa e convertire i valori in secondi e il totale dei secondi in sessantesimi di secondo.

Modulo 1.3.8: Linee 8000-8120

```
8000 REM*********************************
8010 REM STIMA DEL TEMPO
8020 REM*********************************
8030 :
8040 :
8050 TT=0
```

```
8060 TT=TT+3600*VAL(LEFT$(TT$,2))
8070 TT=TT+60*VAL(MID$(TT$,3,2))
8080 TT=TT+VAL(RIGHT$(TT$,2))
8090 TT=TT*60
8100 RETURN
8110 :
8120 :
```

Verifica

Battete:

TT$="123456"[RETURN]
GOTO8000[RETURN]

e il programma si fermerà con il messaggio RETURN WITHOUT GOSUB. Ora battete:

PRINT TT[RETURN]

e il numero che vedrete comparire sarà 2717760.

Modulo 1.3.9: Regolazione dei timer

Ora che possiamo riprodurre lo stato dei timer e tradurre le stringhe in valori cronologici espressi in sessantesimi di secondo, possiamo procedere con il modulo che ci permette di regolare i nostri 10 timer.
Come potete notare, per ottenere i tre valori di cui ha bisogno, il modulo utilizza l'istruzione INPUT, così mentre è in funzione, i timer non vengono campionati e nessuno genera il proprio segnale acustico, finché il modulo non è stato completato.

Modulo 1.3.9: Linee 6000-6130

```
6000 REM*********************************
6010 REM REGOLAZIONE DEL TIMER
6020 REM*********************************
6030 :
6040 :
6050 PART=1 : GOSUB 5080
6055 TT=TI : GOSUB 7000 : CHAR ,14,8,TT$
6060 CHAR ,0,19,""
6065 GET T$ : INPUT "[CD][CD]QUALE TIMER USI
(0-9):";NN
6070 INPUT "FRA QUANTO TEMPO (HHMMSS):";TT$
6080 INPUT "PER QUALE SCOPO:";TT$(NN)
```

```
6090 TI=TI : GOSUB 8000
6100 TT(NN)=TT+TI
6110 IF TT(NN)>5184000 THEN
TT(NN)=TT(NN)-5184000
6120 RETURN
6130 :
6140 :
```

Commento

Linea 6050: Della variabile PART abbiamo già parlato. Il suo scopo è di garantire che il modulo di visualizzazione dei timer ritorni a questo modulo non appena le ore su cui sono regolati i timer sono state riprodotte sullo schermo.

Linea 6065: Osservate la strana istruzione GET all'inizio di questa linea. La sua funzione è di proteggere contro l'eventuale pressione del tasto RETURN da parte dell'utente dopo la scelta nel menu. Se l'istruzione GET non ci fosse e venisse premuto il tasto RETURN, quest'ultimo verrebbe salvato e trattato come se fosse la risposta al primo sollecito del sistema. L'istruzione GET serve ad annullare qualsiasi tasto che sia stato premuto prima della comparsa del sollecito. Se premete due tasti, comunque, avrete i vostri bei problemi.

Linee 6070 e 6090-6100: Il timer viene regolato specificando il periodo di funzionamento prima che venga annullato il comando di emissione del segnale acustico. Il valore numerico viene tradotto in sessantesimi di secondo e il risultato viene aggiunto al contenuto corrente di TI (l'ora corrente espressa in sessantesimi di secondo). Se l'ora che risulta è maggiore di 24 viene ridotta di 24 ore e il risultato è una cifra che rappresenta l'ora in cui deve iniziare l'emissione del segnale acustico del timer specificato dall'utente. Il valore viene quindi memorizzato in un elemento del vettore TT.

Verifica

A questo punto dovreste essere in grado di sottoporre il modulo a verifica eseguendo (RUN) l'intero programma. Non appena avete il menu sullo schermo, scegliete l'opzione 1 e battete:

0,000100 e TEST

in risposta ai tre solleciti del programma, dopodichè il programma ritorna al menu. Ora scegliete l'opzione due che prevede la riproduzione sullo schermo dei timer e vedrete che al timer zero è stato attribuito un valore pari a poco meno di un minuto oltre l'ora al momento (di quanto oltre, dipende dalla vostra rapidità) più l'etichetta TEST. Non attendete il segnale: non avete ancora introdotto il modulo che permette di ottenerlo.

Modulo 1.3.10: Generazione del segnale acustico

A questo punto abbiamo tutti gli elementi del programma funzionanti, salvo quello che permette al timer di emettere il segnale acustico.
Questo modulo e il prossimo aggiungono il tocco finale. Questo produce il suono, l'altro collega il segnale al resto del sistema.

Modulo 1.3.10: Linee 3000-3240

```
3000 REM*********************************
3010 REM SUONERIA
3020 REM*********************************
3030 :
3040 :
3050 GOSUB 14000: COLOR 1,6,4 : TM=1
3060 CHAR ,15,5,"SUONERIA" : PRINT
3070 PRINT "[CD][CD][CD][CD][CD]";I;" - ";
3080 TT=TT(I)
3090 GOSUB 7000
3100 PRINT TT$;" - ";TT$(I)
3110 :
3120 :
3130 VOL 7
3140 TT=TI
3150 T$="" : DO UNTIL TI>TT+3600 OR T$<>""
3160 SOUND 1,750,30
3170 SOUND 1,650,30
3180 FOR J=1 TO 2000 : NEXT
3190 GET T$
3200 LOOP
3210 VOL 0
3220 RETURN
3230 :
3240 :
```

Commento

Linee 3050-3100: Queste linee visualizzano il numero, l'ora di scadenza e l'etichetta del timer che ha richiesto il segnale acustico. Esse inoltre assegnano il valore uno alla variabile TM, ad indicare alle altre parti del programma che un segnale è appena stato attivato.

Linee 3130-3240: Queste linee provocano l'emissione di un segnale a due tonalità della durata di un minuto. Notate che il ciclo che produce il segnale è

preceduto da un comando che alza il volume al massimo, seguito da un altro che lo riduce a zero. Il ciclo ha due elementi di temporizzazione incorporati e continua a produrre il suono finché il valore memorizzato nella variabile di sistema TI è uguale a 3600 più il valore di TT, che contiene il valore del timer corrente. Poiché TI e TT registrano i valori di tempo in sessantesimi di secondo, significa che il ciclo dura un minuto. La seconda forma di temporizzazione è rappresentata dal semplice ciclo di ritardo alla linea 3180 che inserisce uno spazio tra ogni serie di segnali. Nel suo complesso, il ciclo è studiato in modo tale che basti premere un tasto per interrompere l'emissione sonora.

Verifica

Battete:

```
TT =TI[RETURN]
GOTO 3000
```

e otterrete come risposta l'emissione del segnale acustico per la durata di un minuto circa, dopo il quale il programma si fermerà e vedrete comparire il messaggio RETURN WITHOUT GOSUB. Se lo desiderate, potete ripetere la prova, interrompendo il segnale con un tasto qualsiasi. Il messaggio di errore può non apparire subito per il fatto che le note e il ciclo di temporizzazione devono finire prima, ma non è necessario tenere il dito sul tasto, una sola breve pressione dovrebbe essere più che sufficiente.

Modulo 1.3.11: Campionamento dei timer

Nel commento al programma abbiamo accennato al campionamento costante dei timer anche nei momenti in cui il programma sembra non far nulla di particolare. Ebbene, quest'operazione continuata nel tempo è eseguita dal modulo di cui stiamo per parlarvi.

Modulo 1.3.11: Linee 4000-4100

```
4000 REM*********************************
4010 REM MODULO DI CONTROLLO
4020 REM*********************************
4030 :
4040 :
4050 FOR I=0 TO 9
4055 T2=TT(I)
4060 IF T2<TI AND T2>TI-36000 AND T2<>0 THEN
GOSUB 3000 : TT(I)=0 : TT$(I)=""
4070 NEXT I
4080 RETURN
4090 :
4100 :
```

Commento

Linee 4050-4080: L'unica funzione di questo ciclo, che viene continuamente chiamato dalle altre parti del programma, è di esaminare i dieci timer e vedere se l'ora al momento ha superato quella su cui è regolato ciascuno. I segnali acustici vengono attivati solo se mancano meno di 10 minuti al momento in cui dovrebbero entrare in azione. Questo perché, se ad esempio fossero le 2359 (un minuto prima di mezzanotte) e voleste regolare il time sulle 0800 di domani mattina, non sarebbe di grande utilità se si spegnesse nel momento in cui è stato fissato perché 0800 è inferiore a 2400. Supponendo che regoliate il timer per un'ora che è più di dieci minuti prima dell'ora corrente, il programma da' per scontato che l'ora si riferisca al giorno successivo. Il periodo di dieci minuti serve a garantire che, se il programma stà facendo qualcosa in cui i timer non vengono campionati, il segnale acustico non vada perso. Una volta che il segnale acustico è stato attivato, il timer si azzera automaticamente e non suona più.

Verifica

Seguite la procedura di verifica fornita per il modulo 3.8. Questa volta, però, non appena compare sullo schermo l'ora fissata per il timer zero, il segnale acustico deve suonare.

Modulo 1.3.12: Azzeramento dello schermo

Lo scopo di questo programma è di lavorare in sottofondo in modo da agire come sistema di misurazione cronologica e di sveglia. Lasciare il C16 acceso non comporta nessun inconveniente per la macchina; più problematico è invece lasciare sempre acceso il televisore, in quanto l'avere sullo schermo sempre la stessa immagine può far sì che le parti più luminose dell'immagine visualizzata si imprimano leggermente sullo schermo in modo da apparire come macchie permanenti su tutto quanto vi compare poi. (Niente paura, comunque: non stiamo parlando di quei casi in cui vi può capitare di lasciare il televisore acceso per ore, ma di casi in cui la stessa immagine rimane sullo schermo per giorni e giorni!).
Per ovviare a questo inconveniente, il programma dispone di un proprio modulo di protezione dello schermo che azzera quest'ultimo e visualizza un breve messaggio a lettere lampeggianti che, a differenza di quelle permanenti, non si imprimono.
Mentre lo schermo rimane azzerato, i timer continuano ad essere campionati dal modulo precedente e tutti i segnali acustici che arrivano a scadenza vengono attivati nel solito modo.

Modulo 1.3.12: Linee 13000-13120

```
13000 REM*******************************
13010 REM SCHERMO VUOTO E STATO D'ATTESA
13020 REM*******************************
```

```
13030 :
13040 :
13050 T$="" : DO UNTIL T$<>""
13060 SCNCLR : CHAR ,17,12,"[FLASH ON]TIMER":
PRINT
13070 CHAR ,13,14,""
13080 GOSUB 15000
13090 LOOP
13100 RETURN
13110 :
13120 :
```

Verifica

Regolate uno o due timer su un'ora ravvicinata e scegliete nel menu lo stato di schermo azzerato. Anche se non riuscite a vedere in che condizioni sono i timer, vi accorgerete che i segnali acustici vengono attivati normalmente.

Programma 1.4; Dayplan (agenda)

Dopo aver introdotto e sottoposto a verifica il programma dei timer, proveremo ora ad aggiungere alle funzioni che è già in grado di svolgere una serie di nuove che gli consentano di funzionare come agenda, permettendo all'utente di memorizzare l'orario di 16 appuntamenti al giorno per sette giorni, visualizzandoli uno per uno sullo schermo.

Nel far ciò, dimostreremo, come abbiamo già fatto, i vantaggi della programmazione modulare quando si debba potenziare un programma già funzionante.

Poiché lavoreremo su un programma già esistente, elencheremo qui solo i moduli nuovi o modificati. Nel creare il nuovo programma, potremmo usare il vecchio programma TIMER, ma è più sicuro salvarlo (SAVE) come programma DIARY in modo da conservarne la versione originale.

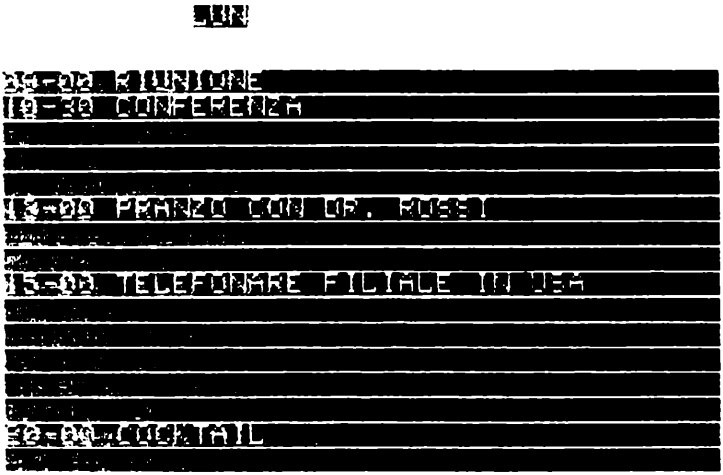
N.B.: Prima di iniziare l'inserimento di questi moduli, controllate di avere in macchina il programma TIMER completo.

Modulo 1.4.1: Inizializzazione

Le nuove capacità della seconda versione del programma richiedono qualche informazione in più, per quando verrà eseguito il primo comando RUN, e il dimensionamento di un vettore per la conservazione dei dettagli relativi ai vari appuntamenti.

Di nuovo ci sono solo le linee 1060, 1070 e 1240-1350, che preferiamo mostrarvi nell'ambito del modulo nella sua interezza.

Figura 13



Modulo 1.4.1: Linee 1000-1350

```
1000 REM*********************************
1010 REM INIZIALIZZAZIONE
1020 REM*********************************
1030 :
1040 :
1050 GOSUB 14000 : COLOR 1,6,4
1060 SP$="
  " : REM 38 SPAZI VUOTI
1070 DIM TT$(9),TT(9),ARRAY$(6,15)
1080 :
1090 :
1100 Q$="" : DO UNTIL Q$="S"
1110 TRAP 11030
1120 INPUT "[CD][CD]INSERISCI LE ORE
(HHMMSS):";TI$
1130 TRAP
1140 TT=TI
1150 GOSUB 7000
```

```
1160 PRINT "[CD]SONO LE ORE: ";TT$
1170 INPUT "[CD]SONO ESATTE (S/N):";Q$
1180 LOOP
1190 :
1200 :
1210 RESTORE : FOR I=0 TO 6 : READ T$ :
DD$(I)=T$ : NEXT
1220 :
1230 :
1240 INPUT "[CD][CD][CD]DEVI CARICARE DA DISCO
(S/N):";LD$
1250 IF LD$="S" THEN GOSUB 12110
1260 :
1270 :
1280 FOR I=0 TO 6 : FOR J=0 TO 15
1290 IF ARRAY$(I,J)="" THEN ARRAY$(I,J)=SP$
1300 NEXT J,I
1310 :
1320 :
1330 DATA DOM,LUN,MAR,MER,GIO,VEN,SAB
1340 :
1350 :
```

Commento

Linee 1050-1060: SP$ viene usata per formattare lo schermo, mentre ARRAY$ conserverà i dati relativi agli appuntamenti e DD$, i nomi dei giorni della settimana. Gli altri due vettori sono già stati usati nel programma TIMER.

Linee 1210-1330: Queste linee permettono la lettura nel vettore DD$ della forma abbreviata dei giorni della settimana.

Linee 1240-1250: La nuova versione del programma dovrà riservare su disco o su nastro memoria sufficiente per i dati relativi agli appuntamenti (vedi il prossimo modulo).

Linee 1280-1300: Queste linee riempiono ogni elemento vuoto del vettore con un numero standard di spazi. Più tardi, questo ci consentirà di formattare lo schermo per visualizzare i vari impegni.

Verifica

Impartite il comando RUN e fornite l'ora. Battete poi N in risposta al sollecito del caricamento da disco e, dopo un breve intervallo, dovreste vedere comparire il menu.

Questo è l'unico tipo di controllo che si può fare a questo stadio.

Modulo 1.4.2: Menu

Il modulo menu deve essere modificato per rendere possibili le nuove funzioni previste. Qui di nuovo ci sono le linee 2120-2150 e 2220.

Modulo 1.4.2: Linee 2000-2280

```
2000 REM*********************************
2010 REM MENU
2020 REM*********************************
2030 :
2040 DO UNTIL Z=7

2050 GOSUB 14000
2060 COLOR 1,12,4
2070 PRINT "[CD][CD][CD]COMANDI DISPONIBILI:"
2080 COLOR 1,4,4
2090 PRINT "[CD][CD]   1) MODIFICA IL TIMER"
2100 PRINT "   2) VISUALIZZA IL TIMER"
2110 PRINT "   3) SCHERMO VUOTO ED ATTESA"
2120 PRINT "   4) MODIFICA GLI IMPEGNI"
2130 PRINT "   5) VISUALIZZA GLI IMPEGNI"
2140 PRINT "   6) REGISTRA GLI IMPEGNI"
2150 PRINT "   7) STOP"
2160 COLOR 1,12,4
2170 PRINT "[CD][CD]LA TUA SCELTA: ";
2180 :
2190 :
2200 GOSUB 15000
2210 Z=VAL(T$)
2220 ON Z GOSUB
6000,5000,13000,10000,9000,12000
2230 LOOP
2240 :
2250 :
2260 SCNCLR : END
2270 :
2280 :
```

Verifica

A questo punto dovreste essere in grado di ottenere il menu con l'inclusione delle

nuove capacità del programma. Il tentativo di accedere ad esse, però, da' come risultato la generazione del messaggio di errore UNDEFINED LINE.

Modulo 1.4.3: Messaggi di errore

Rispetto al modulo originale, la versione potenziata contiene due messaggi di errore in più e, in particolare, le linee 11060-11140.

Modulo 1.4.3: Linee 11000-11140

```
11000 REM*******************************
11010 REM MESSAGGIO D'ERRORE
11020 REM*******************************
11030 PRINT "ORA INSERITA NON CORRETTAMENTE" :
RESUME
11040 :
11050 :
11060 RIGHT=0
11070 PART=1 : SCNCLR : GOSUB 9170
11080 CHAR ,0,23,"TROPPO LUNGO" : FOR I=1 TO
2000 : NEXT
11090 RETURN
11100 :
11110 :
11120 RIGHT=0 : PRINT "GIORNO NON INSERITO
CORRETTAMENTE" : RETURN
11130 :
11140 :
```

Commento

Linee 11060-11090: Queste linee, che non potete ancora usare, sono caratteristiche del tipo di routine necessaria a fornire un messaggio di errore quando non venga usata la funzione TRAP. La routine GOSUB utilizza una parte del modulo successivo per riprodurre di nuovi i dati in forma ordinata prima di dare all'utente la possibilità di introdurre altri dati significativi.

Verifica

Il modulo non può essere sottoposto a verifica finché non è stato introdotto quello di cui stiamo per parlare ora.

Modulo 1.4.4: Visualizzazione degli appuntamenti del giorno

Come nella versione originale del programma TIMER, introdurremo un modulo

di visualizzazione prima del modulo che si permetterà di generare qualsiasi cosa sullo schermo. Anche se per ora non potremo eseguire nessuna prova, dobbiamo poter vedere che il programma è in grado di generare un'immagine ordinata dei dati.

In questo modulo, tutte le linee sono nuove.

Modulo 1.4.4: Linee 9000-9360

```
9000 REM*********************************
9010 REM VISUALIZZA GLI IMPEGNI
9020 REM*********************************
9030 :
9040 :
9050 T$="" : DO UNTIL T$<>""
9060 :
9070 :
9080 GOSUB 14000
9090 :
9100 :
9110 RIGHT=0 : DO UNTIL RIGHT
9120 INPUT "[CD][CD]QUALE GIORNO
(LUN,MAR..):";DD$
9130 RIGHT=1 : FOR DAY=0 TO 6 : IF
DD$<>DD$(DAY) THEN NEXT DAY : GOSUB 11120
9140 LOOP
9150 :
9160 :
9170 GOSUB 14000 : COLOR 1,6,4
9180 CHAR ,17,4,"[RVS ON]" : PRINT DD$(DAY)
9190 FOR J=0 TO 15 : COLOR 1,3-(J AND 1),4
9200 CHAR ,1,J+6,"[RVS ON]"
9210 PRINT ARRAY$(DAY,J)
9220 NEXT J
9230 COLOR 1,6,4
9240 :
9260 IF PART=1 THEN PART=0 : RETURN
9270 :
9280 :
9290 CHAR ,0,23,""
9300 GOSUB 15000
9310 :
9320 :
```

```
9330 LOOP
9340 RETURN
9350 :
9360 :
```

Commento

Linee 9050 e 9330: Questo ciclo riproduce di nuovo i dati sullo schermo ogni volta che essi sono influenzati dalla visualizzazione di un segnale acustico. La riproduzione dei dati può essere interrotta in qualsiasi momento con la semplice pressione di un tasto.

Linee 9110-9140: Queste linee sollecitano l'utente a battere il nome del giorno della settimana in tre lettere, come riportato nella linea 1330 (qualsiasi altro formato verrebbe rifiutato, accompagnato da un messaggio di errore).
L'invocazione del messaggio di errore si basa sul fatto che, se l'utente introduce il nome di un giorno che non è conservato in DD$, non si raggiungerà mai la fine del ciclo alla linea 9130. L'istruzione NEXT viene raggiunta solo se (IF) quello che l'utente ha introdotto non è uguale a ciò che si trova nel vettore DD$. Se il messaggio di errore viene invocato la variabile RIGHT viene azzerata e il ciclo ripetuto.

Linee 9170-9230: Le prime due linee visualizzano il titolo del programma e il giorno della settimana, mentre il ciclo principale, con la variabile J, genera le sedici linee del vettore, che si riferiscono al giorno indicato, proponendole nei colori grigio (forse è meglio dire bianco scuro) e rosso alternati, che vengono stabiliti con l'uso di AND, esattamente come nel modulo di visualizzazione del programma TIMER come noterete, le linee vengono stampate in negativo in modo che tutte le lettere possano risultare leggibili sul fondo dello schermo che è nero. Riempire gli elementi del vettore con spazi, nel primo modulo, fa' sì che le linee colorate siano tutte della stessa lunghezza e risultino nell'insieme più piacevoli.

Linea 9260: Come nel programma TIMER, la variabile PART viene usata per stabilire che il modulo di visualizzazione è invocato da un'altra parte del programma e che non deve quindi andare a mettersi in attesa alla fine del modulo.

Linee 9290-9300: Anche qui, come nel programma TIMER, i 10 timer continuano ad essere tenuti sotto controllo per tutto il tempo durante il quale l'immagine rimane sullo schermo.

Verifica

Lanciate il comando RUN e, appena compare il menu sullo schermo, scegliete l'opzione cinque. Battete un giorno della settimana e vedrete comparire l'immagine ordinata cui abbiamo accennato nell'introduzione a questo modulo, con la parola SONO IN ATTESA lampeggiante sul bordo inferiore dello schermo. Premete

quindi un tasto qualsiasi, ritornando al menu e, dopo aver scelto ancora l'opzione cinque, non appena vi viene chiesto di battere un giorno della settimana, provate a introdurre un dato sbagliato per vedere se, in questo caso, compare un messaggio di errore.

Modulo 1.4.5: Introduzione dei dati relativi agli appuntamenti

Poiché siamo ora in grado di visualizzare i nostri impegni, possiamo passare alla routine che ci permette di cancellarli o di introdurne i dati relativi.

Modulo 1.4.5: Linee 10000-10460

```
10000 REM*******************************
10010 REM MODIFICA IMPEGNI
10020 REM*******************************
10030 :
10040 :
10050 PART=1
10060 GOSUB 3080
10070 :
10080 :
10090 QUIT$="" : DO UNTIL QUIT$="N"
10100 :
10110 :
10120 NN=16 : DO UNTIL NN>=0 AND NN<=15
10130 CHAR ,0,23,"POSIZIONE DEL NUMERO:" :
INPUT NN
10140 LOOP
10150 :
10160 :
10170 TT$="" : DO UNTIL LEN(TT$)=6 : TT$=""
10180 CHAR ,0,23,SP$
10190 CHAR ,0,23,"ORA (HHMM):" : INPUT TT$ :
TT$=TT$+"00"
10200 LOOP
10210 CHAR ,0,23,SP$
10220 GOSUB 8000 : GOSUB 7000
10230 :
10240 :
10250 RIGHT=0 : DO UNTIL RIGHT : RIGHT=1
10260 CHAR ,0,23,"IMPEGNO:" : INPUT Q$
10270 IF LEN(D$)>28 THEN GOSUB 11060
```

```
10290 :
10300 :
10310 ARRAY$(DAY,NN)=LEFT$(LEFT$(TT$,5)+"
"+Q$+SP$,38)
10320 IF TT=0 THEN ARRAY$ (DAY,NN)=SP$
10330 :
10340 :
10350 CHAR ,0,23,SP$
10360 PART=1 : GOSUB 9180
10370 CHAR ,0,23,SP$
10380 :
10390 :
10400 CHAR ,0,23,"UN ALTRO IMPEGNO (S/N):"
10410 GOSUB 15000
10420 QUIT$=T$
10430 LOOP
10440 RETURN
10450 :
10460 :
```

Commento

Linee 10050-10060: Il modulo è usato per ricavare il giorno della settimana da visualizzare e per poter leggere le condizioni dell'appuntamento registrato. La variabile PART ha lo scopo di assicurare che l'esecuzione ritorni ad esso non appena sullo schermo sono comparsi i dati.

Linee 10090 e **10400-10430:** Il modulo richiede la pressione di un tasto.

Linee 10120-10140: I dati relativi ad ogni singolo appuntamento possono essere inseriti in 16 posizioni numerate da 0 a 15. Alla variabile, che viene usata per specificare la posizione in cui va collocato un certo dato, viene prima assegnato il valore 16, in modo che sia fuori limite, e il ciclo continuerà a sollecitare l'utente finché non batterà un numero valido. Questo risolve il problema dell'utente che può premere semplicemente RETURN, confermando l'ultimo valore introdotto della variabile. Se non specificassimo il valore di NN all'inizio del ciclo, i dati potrebbero essere introdotti in un punto non desiderato. Con NN=16, il ciclo si limita a ripetere il sollecito ogni volta che viene premuto il tasto RETURN.

Linee 10170-10220: Il ciclo azzera la linea in basso generando SP$ e poi accetta dall'utente un dato in ingresso, costituito dall'ora dell'impegno. Il tempo è introdotto in ore e minuti, senza i secondi che sono sostituiti dalle cifre 00, aggiunte per poter ottenere il normale formato di sei caratteri. Il ciclo DO garantisce che il sollecito sia ripetuto finché non vengono introdotti quattro caratteri. La stringa

così ottenuta viene inviata al modulo che la traduce in un numero di sessantesimi di secondo e poi a quello che trasforma la nuova rappresentazione in una stringa nella forma 00-MM-SS (ore, minuti, secondi). Come potete notare, non importa se l'utente introduce un'ora in un formato non valido, purché di quattro caratteri: le funzioni cronologiche interne del C16 non vengono influenzate e quindi il programma non subisce interruzioni. Un'eventuale ora non valida viene probabilmente tradotta in 00-00-00 e non viene acquisita (vedi commento alle linee 10310-10320).

Linee 10250-10280: La dicitura dell'impegno viene introdotta in questo punto, accompagnata da un messaggio di errore e il sollecito viene ripetuto se il dato in ingresso è più lungo di 28 caratteri.

Linee 10310-10320: Queste linee collocano ciò che è stato introdotto dall'utente nel vettore ARRAY$ (o annullano il contenuto corrente di una certa posizione se l'ora battuta era 0000 (questo è il modo in cui i dati vengono cancellati). La posizione del vettore è stabilita in parte da NN, che è la posizione specifica all'inizio di questo modulo, in parte da DAY, che era il valore cui si è arrivati con il ciclo alla linea 9130 del modulo precedente. L'espressione di stringa è un formato standard, con spazi che vengono aggiunti alla fine per completarla e di cui vengono usati i primi 38 caratteri. Ogni serie di dati in ingresso viene quindi ad essere della stessa lunghezza delle celle "vuote" del vettore, che contengono tutte 38 spazi.

Linee 10350-10370: Poiché uno dei dati in ingresso è cambiato, lo schermo viene modificato per riflettere la variazione.

Verifica

Battete il comando RUN e scegliete l'opzione 4 del menu. Alle tre richieste del computer circa la posizione, l'ora e l'impegno, battete:

0/0830/123456789

In risposta, vedrete comparire l'ora e l'impegno introdotti nella prima posizione. Al sollecito UN ALTRO IMPEGNO? generato dal computer, rispondete battendo Y e introducete un altro dato con i seguenti dettagli:

1/0930/12345678901234567890123456789

Questa volta, non appena ricompaiono i dati sullo schermo, non vi troverete quelli appena battuti, ma il messaggio TOO LONG (troppo lungo) visualizzato sul bordo inferiore dello schermo.

Modulo 1.4.6: Memorizzazione dei dati su disco o nastro

Possiamo ora passare per la prima volta ad un campo che molti trascurano a loro spese, vale a dire la memorizzazione dei dati su nastro o su disco (secondo il tipo di hardware installato). Poiché avrete senza dubbio scoperto di avere

commesso degli errori nel battere i nostri programmi, siamo certi che saprete quanto è fastidioso dover introdurre tutti i dati di nuovo. Inoltre è ben poco utile avere un programma che tiene la propria agenda e doverlo ribattere ogni volta che si spegne il computer. Forse questo programma non crea problemi, ma come pensate vi sentireste a dover battere centinaia di informazioni?
Il problema può essere risolto con l'uso di un drive o di una cassetta su cui i dati introdotti possono essere conservati in modo permanente e poi richiamati nel C16 ogni volta che se ne deve fare uso. La gestione del disco o del nastro è affidata a questo modulo.

Modulo 1.4.6: Linee 12000-12160

```
12000 REM*******************************
12010 REM FILES DATI SU DISKETTE
12020 REM*******************************
12030 :
12040 :
12050 OPEN 1,8,2,"@0:TIMEDATA,S,W"
12060 FOR I=0 TO 6 : FOR J=0 TO 15
12070 PRINT#1,ARRAY$(I,J) : NEXT J,I
12080 PRINT#1 : CLOSE1 : RETURN
12090 :
12100 :
12110 OPEN 1,8,2,"TIMEDATA,S,R"
12120 FOR I=0 TO 6 : FOR J=0 TO 15
12130 INPUT#1,ARRAY$(I,J) : NEXT J,I
12140 CLOSE1 : RETURN
12150 :
12160 :
```

Commento

Linee 12050-12080: Queste linee memorizzano su disco i dati introdotti nel programma (le informazioni relative al nastro verranno esaminate alla fine del modulo).

Linea 12050: Prima che i dati siano trasferiti sul disco, occorre predisporne lo spazio ad essi destinato con un'operazione chiamata "apertura del file" che viene eseguita con il comando

```
OPEN<FILE>,<DEVICE>,<CHANNEL>,"[@0:]<FILENAME>,S,W"
  (1)    (2)      (3)        (4)       (5)       (6)      (7)(8)
```

1) L'istruzione inizia con il comando OPEN che dice al drive quale operazione gli viene chiesto di eseguire.
2) Il numero del file, di solito fra 1 e 27, inclusi. È consentito avere più file aperti allo stesso tempo per scopi diversi, ma ognuno deve avere un numero suo.
3) Ogni dispositivo di memorizzazione o di stampa collegato al C16 ha un suo numero di identificazione in modo da riconoscere una richiesta di intervento indirizzatagli. Nel caso in cui sia collegato un solo drive, il numero di dispositivo è nella maggior parte dei casi otto.
4) A parte i diversi numeri di file, è necessario specificare quale delle diverse linee di comunicazione fra il C16 e il drive verrà impiegata nell'operazione specificata nella linea di comando. Le linee che vengono di solito usate sono quelle comprese fra 2 e 14 e di norma si può usare una sola linea per file alla volta.
5) Le parentesi quadre indicano che i caratteri racchiusi sono facoltativi. Abbiamo già parlato della sequenza @0: quando abbiamo parlato del salvataggio dei programmi con l'istruzione SAVE. La sua funzione in questo caso è la stessa: scrivere sopra un file già esistente e con lo stesso nome, se esiste naturalmente. Nei programmi di grandi dimensioni, vedremo in che modo è possibile modificare il nome di un file in uso in modo che lo stesso programma possa memorizzare i diversi file di dati sullo stesso disco. Per il momento, questo programma cancella tutti i dati precedentemente memorizzati sul disco ogni volta che ne viene trasferito uno nuovo.
6) Come il nome del programma, il nome del file può essere costituito da non più di 16 caratteri.
7) Il drive è in grado di trattare diversi tipi di file, ciascuno con una propria funzionalità ben distinta. Per questa applicazione, useremo ciò che è noto come "file sequenziale", indicato dalla lettera S. La caratteristica distintiva di questo tipo di file è che i dati vengono memorizzati uno dopo l'altro e possono essere letti nello stesso ordine in cui sono stati memorizzati.
8) La lettera W indica che il file è stato aperto per un'operazione di memorizzazione (in inglese Write = scrivere su disco) di dati su disco e non per un'operazione di richiamo.

Linee 12060-12070: Dopo aver aperto il file con il comando OPEN, non ci rimane che trasferirvi le informazioni. L'operazione viene eseguita con il comando speciale PRINT$. Una cosa importante da notare è che non può essere sostituito dal punto interrogativo. Se lo faceste, sembrerebbe perfetto nel listato, ma produrrebbe un errore di sintassi.
I dati trasferiti nel file sono costituiti dal contenuto del vettore principale (ARRAY$) che contiene i dati relativi agli impegni dell'agenda. Qualcosa in più bisognerebbe dire approposito dei segni di punteggiatura che vanno inseriti tra le voci quando vengono stampate mediante l'uso di un unico comando PRINT$, ma siccome ogni ripetizione del ciclo di questo modulo stampa solo una voce, rimandiamo le spiegazioni al secondo capitolo.

Linea 12080: Durante il trasferimento di dati su disco è consigliabile terminare con un'istruzione PRINT # <NUMERO DI FILE> che garantisca che tutte le voci ancora nella memoria del C16, in attesa di essere stampate, sono state cancellate. Per finire, bisogna chiudere il file, usando il comando CLOSE. Il non farlo fa' sì che il numero del file non sia più disponibile per altri impieghi e, cosa ancora peggiore, può provocare una perdita dei dati sul disco.

Linee 12110-12140: Queste linee hanno la funzione opposta rispetto a quelle appena commentate, nel senso che richiamano dal disco dati precedentemente memorizzati.

Linea 12110: Ancora una volta bisogna aprire (OPEN) un file. La sola differenza fra questa e l'istruzione precedente è che questa non obbliga all'uso dei caratteri "@0:" e la lettera W è sostituita dalla lettera R che sta' per "read" (leggere).

Linee 12120-12130: L'istruzione opposta a PRINT# è INPUT# (anche in questo caso non possiamo usare l'abbreviazione standard di INPUT).

Linea 12140: Come per la linea 12080, il file deve essere sempre chiuso.

Verifica

(chi usa il nastro legga attentamente quanto segue)

Battete il comando RUN e introducete alcune date relative ad altrettanti appuntamenti e impegni di una vostra ipotetica agenda. Tornate al menu e scegliere l'opzione del salvataggio (SAVE).
Non appena ritorna il sollecito del menu, interrompete l'esecuzione del programma. A questo punto potete battere il comando RUN di nuovo e questa volta rispondere Y al programma che vi chiede se volete eseguire un caricamento da disco. Ora potete essere certi che i dati sono stati caricati.

Modifiche per l'uso del nastro

Per adattare il modulo all'uso con la cassetta, occorre apportargli le seguenti modifiche:

1) Modificate la prima istruzione OPEN in modo che si legga:

```
OPEN 1,1,1, "TIMEDATA,S,W"
```

Come potete notare, abbiamo cambiato due cifre. Il numero del dispositivo è uno, il numero corretto per il registratore, e uno è anche il numero del canale, cioè il numero che indica al C16 che i dati vanno salvati su nastro. La sequenza @0: non viene inserita in quanto la cassetta automaticamente scrive i dati su quelli già esistenti.

2) Modificate la seconda istruzione in modo che diventi:

```
OPEN1,1,0,"TIMEDATA,S,R"
```

Ancora il numero del dispositivo è uno, mentre quello del canale è zero. Questo secondo numero indica al C16 che i dati devono essere richiamati dal nastro. Se state lavorando con le cassette, i dati per le diverse agende possono essere memorizzati tenendo ogni file in diverse posizioni del nastro o addirittura su nastri diversi.
Gli altri comandi del modulo vanno bene sia con i dischi che con i nastri.

CAPITOLO 2

Dipingere con i numeri

In questo capitolo ci spostiamo dai giochi con l'ora e con la misurazione del tempo in genere a qualcosa che il C16 è in grado di fare particolarmente bene, cioè visualizzare le informazioni in modi più facilmente comprensibili che non attraverso la descrizione dei fatti o l'uso dei numeri. Il capitolo presenta tre programmi che vi permetteranno di creare grafici di diverso tipo, sia a bassa che ad alta risoluzione.
Come nel capitolo che lo ha preceduto e come in quello che lo seguirà, anche in questo scoprirete che le spiegazioni, salvo quelle in cui sono presentate nuove idee o moduli più complessi, diventeranno sempre più corte.
La cosa è voluta ed è la conseguenza del nostro sforzo di trovare un equilibrio fra le informazioni, di cui avete bisogno per capire i programmi, e i programmi stessi, che sono in effetti la ragione per cui avete acquistato questo libro. Se vi accorgete di avere qualche difficoltà a capire quanto vi viene proposto dal programma, fate un passo indietro e cercate qualche programma che utilizza una tecnica simile a quella che non capite, rileggendovi il commento. È per questo motivo che nell'introduzione vi abbiamo chiesto di leggere tutto il libro sistematicamente senza saltare nessun programma.
I programmi di questo secondo capitolo sono:

GRAPH, che crea un grafico ad alta risoluzione molto efficace.

PIECHART, che acquisisce un certo numero di dati e li visualizza in forma di cerchio suddiviso in settori multicolori.

GRAPH II, che permette di creare un meraviglioso diagramma a barre triimensionali.

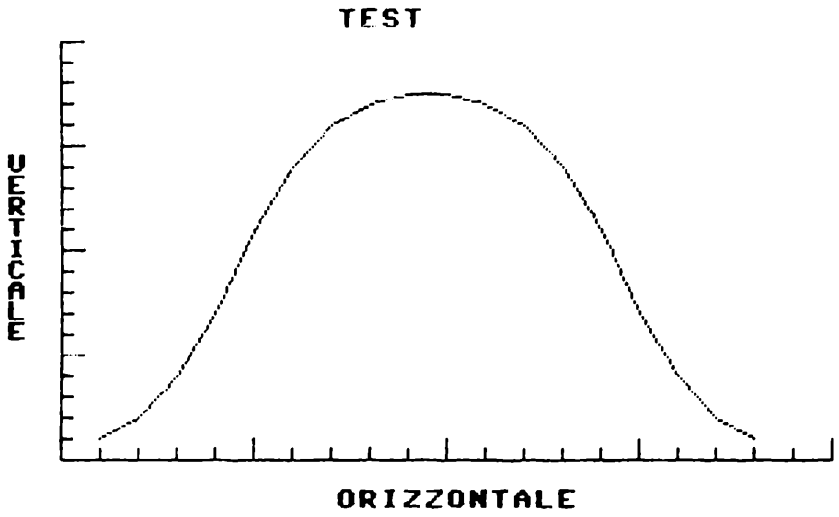
Programma 2.1: GRAPH

Scopi del programma

Uno dei problemi che si incontrano più spesso, lavorando nei limiti di una memoria di 16K, è che i programmi ad alta risoluzione consumano molta della memoria a disposizione dell'utente, lasciandone ben poca per l'elaborazione dei dati dei programmi applicativi. Il programma che stiamo per creare è facile da usare e allo stesso tempo non costringe a destinare memoria per la generazione di messaggi sullo schermo e la memorizzazione di dati nei vettori o su disco o nastro. I programmi interattivi, cioè quelli che chiedono all'utente le informazioni

durante la fase di esecuzione, sono certamente molto piacevoli da usare, ma possono costituire un lusso se la memoria è limitata.
In questo programma, che disegna un grafico ad alta risoluzione, faremo uso di istruzioni DATA, allo scopo di ottenere istruzioni facili da usare e molto più semplici da scrivere di quelle che i programmi interattivi impongono per arrivare allo stesso scopo.

Figura 2.1: Esempio di grafico



Le tecniche utilizzate in questo programma sono:

1) L'uso flessibile delle istruzioni DATA
2) L'uso delle condizioni logiche come variabili.

Moduli 2.1.1 e 2.1.2: I dati per la creazione del grafico

Questo è un modulo che contiene la chiave della semplicità dell'intero programma. All'interno di queste poche linee sono contenute tutte le informazioni necessarie per poter disegnare un grafico soddisfacente, chiaramente etichettato, in modo che l'utente non abbia alcuna difficoltà nel ricrearlo. Non preoccupatevi se a questo livello molti numeri non vi sono del tutto comprensibili. Il loro impiego vi diventerà più chiaro quando avrete introdotto ed eseguito il programma una o due volte e avrete fatto qualche prova apportandovi delle modifiche.

Moduli 2.1.1 e 2.1.2: Linee 3000-4090

```
3000 REM*********************************
3010 REM DATA 1
3020 REM*********************************
3030 DATA TITOLO GRAFICO, TEST
3040 DATA X/ASSE,ORIZZONTALE
3050 DATA Y/ASSE,VERTICALE
3060 DATA UNITA' PER X/ASSE,20
3070 DATA UNITA' PER Y/ASSE,20
3080 DATA VALORE PER V/UNITA',10

4000 REM*********************************
4010 REM DATA 2
4020 REM*********************************
4030 DATA 10,20,40,70,110,140,160,170,175
4040 DATA 175,170,160,140,110,70,40,20,10
4050 DATA FINE
4060 DATA
4070 DATA
4080 DATA
4090 DATA
```

Commento

Linee 3030-3050: Il nome che l'utente intende attribuire al grafico nel suo insieme e le etichette da aggiungere agli assi verticale e orizzontale. Come potete notare, in ogni caso, la frase prima della virgola nell'istruzione DATA e lì solo per comodità dell'utente e verrà ignorata dal programma. La virgola, invece, è essenziale per separare la prima frase dalle informazioni importanti che vengono dopo.

Linee 3060-3070: Per facilità di lettura, i due assi del grafico vengono suddivisi in parti uguali, il cui numero viene specificato dall'utente. Il programma prevede che i due assi siano della stessa lunghezza.

Linea 3080: Se, ad esempio, il grafico fosse progettato per registrare le tonnellate di grano prodotte da un certo paese nel corso di un certo numero di anni, l'utente potrebbe far corrispondere ad ogni singola suddivisione dell'asse verticale un'unità di 1.000 tonnellate.
Invece di far dividere all'utente l'importo effettivo in unità di 1000 prima di introdurre il valore, il numero sulla linea permette di specificare le unità in modo che i numeri possono essere introdotti così come sono.

Linee 4030-4040: I dati su cui è basato il grafico. In questo programma, i numeri producono una linea continua a forma di campana. Come potete osservare, è previsto un solo numero per ciascuna unità dell'asse orizzontale, a partire dalla posizione uno, anche se non è necessario usare l'intero asse. Le linee possono contenere tutte le istruzioni DATA che si desidera purché entro il limite di due linee di schermo per linea di programma.

Linee 4050-4090: Queste linee prevedono l'introduzione di altri dati che in questo caso, comunque, non sono necessari. Ricordate che potete usare tutte le istruzioni DATA consentite dalla memoria, ma è indispensabile che le informazioni introdotte siano chiuse da un'istruzione DATA contenente la parola END, la quale costituisce il segnale con cui il programma viene informato che i dati da usare per il grafico sono finiti, anche se dopo seguono altre istruzioni DATA.

Modulo 2.1.3; Disegno della griglia

Questo modulo ha il compito di disegnare la griglia su cui dovrà essere collocato il grafico, completa delle suddivisioni degli assi e delle varie etichette.

Modulo 2.1.3: Linee 1000-1200

```
1000 REM*********************************
1010 REM DISEGNA GLI ASSI
1020 REM*********************************
1030 COLOR 0,1 : COLOR 1,8,4
1040 GRAPHIC 1,1
1050 LOCATE 20,20
1060 DRAW TO 20,180
1070 DRAW TO 300,180
1080 RESTORE 3000
1090 READ T$,T$ : CHAR ,15,1,T$
1100 READ T$,T$ : CHAR ,15,24,T$
1110 READ T$,T$ : FOR I=1 TO LEN(T$) : CHAR
,0,7+I,MID$(T$,I,1) : NEXT
1120 READ T$,NV : LV=INT(160/NV)
1130 FOR I=1 TO NV
1140 DRAW,20,180-I*LV TO
24-4*(I/5=INT(I/5)),180-I*LV
1150 NEXT
1160 READ T$,NH : LH=INT(280/NH)
1170 FOR I=1 TO NH
1180 DRAW,20+I*LH,180 TO
20+I*LH,176+4*(I/5=INT(I/5))
```

```
1190 NEXT
1200 READ T$,UNITA
```

Commento

Linee 1050-1070: I due assi vengono disegnati in questo modo (approssimativamente): una linea dall'angolo in alto a sinistra a quella in basso sempre a sinistra e poi in senso orizzontale fino all'angolo in basso a destra.

Linea 1080: Il puntatore dati del C16 viene fatto puntare al primo valore di DATA dopo l'avvio del modulo alla linea 6000. L'istruzione RESTORE impedisce la generazione dell'errore di superamento "OUT OF DATA" se il programma inizia con GOTO. L'uso del comando RUN fa' indirizzare comunque il puntatore al primo valore di DATA.

Linee 1090-1110: Vengono lette dalle istruzioni DATA e riprodotte su schermo le etichette del grafico nel suo insieme e quelle degli assi orizzontale e verticale. Nel caso dell'etichetta dell'asse verticale, per riprodurre l'etichetta carattere per carattere lungo il lato sinistro dello schermo, viene utilizzato un ciclo. Notate che, in ogni caso, ci sono due istruzioni READ: la prima prende la frase prima della virgola nell'istruzione DATA. Questa viene poi scaricata leggendo un'altra stringa nella stessa variabile, T$.

Linea 1120: Viene letto dall'istruzione DATA successiva il numero di suddivisioni dell'asse verticale (NV). La lunghezza dell'asse verticale (160 pixel) viene quindi divisa per questo numero in modo da arrivare alla lunghezza in pixel (LV) di ciascuna suddivisione.

Linee 1130-1150: Questo ciclo disegna dei piccoli segni sull'asse verticale in modo da riportare sullo schermo le suddivisioni specificate dall'utente. Una cosa da notare qui è l'espressione 4 * (1/5=INT(1/5)), che ha l'effetto di rendere uno dei segni di ciascuna suddivisione più grande degli altri. Per capirla, è indispensabile conoscere qualcosa sul modo in cui vengono trattate dal C16 le condizioni logiche.
Quando l'interprete BASIC, il grosso programma in codice macchina che esegue il BASIC per voi sul C16 arriva ad una condizione introdotta da un'istruzione IF, cioè qualcosa come A>B, A=B o A < = B, ha bisogno di stabilire se quella condizione è vera o falsa, prima di decidere se eseguire l'azione specificata da quell'istruzione. Così:

```
IF A>B THEN GOSUB 1000
```

avrà effetto se A>B è vero (cioè se, ad esempio, A=10 e B=9), verrà invece ignorata se è falsa (cioè se A=9 e B = 10). Per prendere la decisione, la condizione deve essere valutata in base al valore corrente di A e B (o qualsiasi altra variabile specificata) e le viene assegnato un valore che è meno uno se la condizione è vera e zero se è falsa. È il valore più che la condizione in sé che

conta, un fatto che può essere dimostrato eseguendo una piccola prova. Battete, nel modo diretto:

```
A=5[RETURN]
IF A THEN PRINT "VERO" [RETURN]
```

Ciò comporterà la comparsa sullo schermo della parola VERO dato che il valore successivo a IF non è zero (lo stesso effetto è ottenuto per qualsiasi valore positivo o negativo diverso da zero. Ora provate a battere:

```
A=0[RETURN]
IF A THEN PRINT "VERO" [RETURN]
```

Questa volta sullo schermo non comaprirà nessuna parola. In questo momento, comunque, non ci interessa tanto il modo in cui IF lavora, quando il modo in cui vengono valutate le diverse condizioni. Ora battete:

```
A=1[RETURN]
B=1[RETURN]
PRINT A=B[RETURN]
```

e ciò che dovreste vedere è —1, il valore della condizione vera. Ora battete:

```
A=[RETURN]
B=2[RETURN]
PRINT A=B[RETURN]
```

Il risultato sarà ora zero visto che la condizione non è vera. A prima vista tutto questo può sembrare interessante, ma poco rilevante. In realtà, questa capacità di estrarre un valore da una condizione logica è di estrema importanza in programmazione, come dimostra la linea 1140.
Ciò che questa linea fa' è disegnare una serie di linee perpendicolari all'asse verticale per contrassegnare le divisioni specificate dall'utente. La lunghezza di queste linee è normalmente di quattro pixel, però ogni volta che il valore della variabile I del ciclo è esattamente divisibile per cinque (cioè ogni cinque lineette) la condizione (I/5=INT (I/5)) sarà vera e acquisirà il valore meno uno invece di zero. In altre parole, l'inserimento dell'espressione —4 * (I/5=INT(I/5)) nella linea che specifica la lunghezza della lineetta di suddivisione da disegnare (DRAW), permette di ottenere ogni cinque lineette una di lunghezza doppia senza l'uso di un costrutto IF..THEN..ELSE che risulterebbe molto più complesso e userebbe molto spazio di memoria. Ricordate che per aggiungere quattro alla lunghezza della lineetta dobbiamo togliere quattro volte il valore della condizione, dato che il suo valore, quando è vero, è meno 1 (togliere un numero negativo equivale ad aggiungerne uno positivo).

Linee 1160-1190: Lo stesso procedimento viene seguito per l'asse orizzontale. Notate che in questo caso le lineette devono essere disegnate verticalmente a

partire dall'asse e quindi si muovono sullo schermo dalla posizione 180 alla posizione 176. Per aumentare la lunghezza del segno, 176 deve scendere a 172 e la variazione è ottenuta aggiungendo quattro volte il valore della condizione vera.

Linea 1200: Il numero delle unità rappresentate da ciascuna suddivisione sull'asse verticale viene acquisito dall'istruzione DATA.

Verifica

Per eseguire la verifica di questa parte del programma, tutto ciò che occorre fare è eseguire il programma finora in vostro possesso. Dovreste così vedere disegnarsi la griglia con la parola TEST in alto, VERTICALE lungo l'asse verticale e ORIZZONTALE lungo quello orizzontale. I due assi dovrebbero inoltre risultare suddivisi in 20 parti ciascuno.
Attenzione: alla fine della prova rimarrete nel modo ad alta risoluzione. Non è quindi necessario usare un tasto funzione per eseguire l'istruzione GRAPHIC e aggiungere 0 o tentare di battere GRAPHICO per ottenere di nuovo lo schermo con il testo. Basta premere semplicemente un qualsiasi tasto lettera (non numero), seguito dal tasto RETURN, che genera un errore di sintassi, riportandovi automaticamente nel modo a bassa risoluzione. Questa è una tecnica molto semplice che vi permetterà di risparmiare in futuro un mucchio di tempo.

Modulo 2.1.4: Disegno del grafico

Dopo aver creato la griglia di appoggio, passiamo ora al grafico vero e proprio, usando le informazioni specificate nel modulo DATA2 (Modulo 2.1.2).

Modulo 2.1.4: Linee 2000-2200

```
2000 REM*********************************
2010 REM DISEGNA IL GRAFICO
2020 REM*********************************
2030 TRAP 2180
2040 RESTORE 4000
2050 COL=1
2060 READ T1$,T2$
2070 DO UNTIL T2$="FINE"
2080 READ T3$
2090 X1=20+LH*COL
2100 Y1=180-INT(VAL(T1$)/UNITA*LV)
2110 COL=COL+1
2120 X2=20+LH*COL
2130 Y2=180-INT(VAL(T2$)/UNITA*LV)
```

```
2140 DRAW,X1,Y1 TO X2,Y2
2150 T1$=T2$ :T2$=T3$
2160 LOOP
2170 GETKEY A$
2180 GRAPHIC 0,1
2190 LIST 3000-
2200 END
```

Commento

Linee 2030 e 2170-2190: Premendo STOP in qualsiasi momento durante la sequenza di traccia, o premendo un tasto qualsiasi dopo che il grafico è stato disegnato, si ottiene il ritorno del C16 nel modo a bassa risoluzione e l'elencazione dei dati sui quali è basato il grafico, che consente di esaminare il grafico ottenuto e di modificarne i dati molto facilmente.

Linea 2050: COLUMN viene usato per registrare la posizione di un valore lungo l'asse orizzontale.

Linea 2060: Vengono presi due valori di DATA. La ragione per cui vengono letti (READ) due dati prima del ciclo principale è che il grafico può essere iniziato solo se esiste un punto da cui partire e un altro in cui terminare.

linea 2070; Ad ogni passaggio attraverso il loop, la nuova linea viene tracciata solo se esiste la coordinata valida del punto di destinazione e non in base all'indicatore END.

Linee 2110-2130: Queste linee definiscono i due punti tra i quali vanno disegnati due punti del grafico. La coordinata X, o orizzontale, viene calcolata moltiplicando COLUMN (il numero di unità di cui il grafico è progredito lungo l'asse orizzontale) per la lunghezza in pixel delle unità orizzontali (LH). La costante 20 è la distanza fra l'inizio dell'asse orizzontale e il bordo sinistro dello schermo. La coordinata Y, o verticale, è invece più complessa. Partendo dall'estremità inferiore dell'asse, che è 180 pixel sullo schermo, il valore di DATA viene prima diviso per UNIT. Così, se DATA è uguale a 1.000.000 e l'utente fa', in modo che l'asse verticale sia diviso in unità di 100.000 (UNIT=100.000), il risultato sarà 1.000.000/100.000 o 10 unità. Una volta ottenuto, il numero di unità viene moltiplicato per la lunghezza in pixel delle unità verticali (LV) e, poiché lo schermo è numerato da zero a partire dall'alto, il valore ottenuto viene sottratto a 180.

Linea 2150: Ogni volta che sul grafico viene tracciata una linea, il suo punto di arrivo si trasforma nel punto di partenza di quella successiva. Se però T3$ contiene END, la linea successiva pone fine all'esecuzione del ciclo.

Verifica

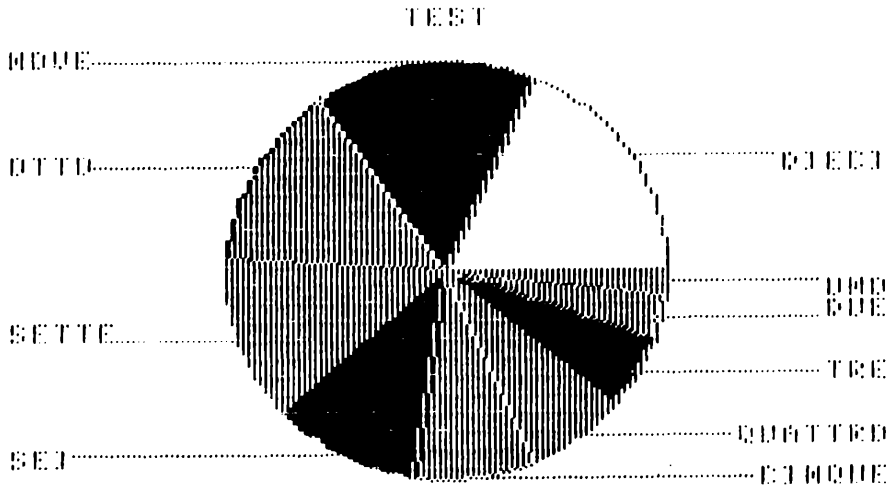
Eseguite il programma e vedrete disegnarsi sullo schermo una linea continua a forma di campana. Quando il disegno è terminato, premete un tasto qualsiasi e vedrete i moduli DATA elencati sullo schermo in modo che, volendo, vi sia possibile modificarli.

Programma 2.2: PIE CHART

Scopi del programma

Un modo molto utile per presentare piccole serie di dati è quello basato sui diagrammi a torta, in cui il totale dei dati disponibili è rappresentato da un cerchio suddiviso in settori che ne rappresentano una percentuale o, più genericamente, una parte. Nel programma che segue utilizzeremo ciò che abbiamo già imparato sulla matematica del cerchio e sull'uso flessibile delle istruzioni DATA. Se ci sono aspetti che non vi sono molto chiari, rileggetevi l'ultimo programma appena visto: non intendiamo infatti soffermaci più su quanto abbiamo già detto.

Figura 2.2: Diagramma torta



(Notate la grossolanità delle parole nel modo grafico due!)

Modulo 2.2.1: I dati per il diagramma

Come nel caso del grafico ad alta risoluzione, i valori numerici su cui è basato il diagramma di questo secondo programma, sono contenuti in istruzioni DATA, così facili da spiegarsi da sè. Notate, tuttavia, che nel programma ,così come

risulta listato, i due vettori che saranno usati per contenere il nome di ciascun dato e il valore di quest'ultimo non sono dimensionati e quindi dovete accontentarvi di un diagramma con non più di 10 settori. A dire il vero, un diagramma a torta con più di dieci valori può essere di poca utilità, perché risulterebbe troppo confuso.
Anche a queste condizioni, volendo, potete comunque includere all'inizio del programma un'istruzione di dimensionamento. Ricordate però che così facendo e aumentando il numero dei valori, potreste accorgervi di dover rinunciare ad alcune etichette dato che il programma raggiunge i limiti imposti dalla memoria.

Modulo 2.2.1: Linee 4000-4080

```
4000 REM*********************************
4010 REM DATI PER IL DIAGRAMMA
4020 REM*********************************
4030 DATA TITOLO,TEST
4040 DATA NUMERO DEI MEMBRI,10
4050 DATA
NOMI,UNO,DUE,TRE,QUATTRO,CINQUE,SEI,SETTE,OTTO
,NOVE,DIECI
4060 DATA
4070 DATA QUANTITA',1,2,3,4,5,6,7,8,3,10
```

Modulo 2.2.2: Elaborazione dei dati per il diagramma

Le informazioni contenute nel modulo DATA vengono lette nelle variabili NAME$ e ITEMS e dei vettori NAME$ e A

Modulo 2.2.2: Linee 5000-5110:

```
5000 REM*********************************
5010 REM PREPARAZIONE DEI DATI
5020 REM*********************************
5030 RESTORE 4000
5040 READ T$,NOME$
5050 READ T$,MEMBRI
5060 READ T$ : FOR I=0 TO MEMBRI-1 : READ
NOME$(I) : NEXT
5070 RESTORE 4070
5080 SOMMA=0 : READ T$ : FOR I=0 TO MEMBRI-1 :
READ T : SOMMA=SOMMA+T : NEXT
5090 RESTORE 4070
```

```
5100 READ T$ : FOR I=0 TO MEMBRI-2 : READ T :
A(I+1)=(T/SOMMA)*360+A(I) : NEXT I
5110 RETURN
```

Linee 5080-5100: I valori dei dati da trasformare in diagramma vengono inseriti prima tutti insieme, per conoscere il totale che deve essere rappresentato dal cerchio. Il puntatore DATA viene quindi riportato (RESTORE) all'inizio dei valori di quantità e ciascuna quantità viene tradotta in una seconda cifra che, divisa per 360, fornisce lo stesso risultato della quantità originale divisa per il totale. Per esempio, se il totale è 100 e la quantità di un dato è 25, il risultato sarebbe 90 cioè il 25% per cento di 360. Queste cifre verranno usate più avanti per stabilire la grandezza dei settori del diagramma a torta attribuiti ai diversi valori.

Verifica

Battete la linea riportate qui di seguito, tratte da ciò che diventerà poi il modulo di controllo ed eseguite il programma:

```
1040 GOSUB 5000
1110 END
```

Se va tutto bene, non dovreste vedere comparire niente: lo schermo riporta qualcosa solo se c'è un errore di qualche tipo.
Se volete, comunque, potete generare il contenuto delle variabili e dei vettori nominati nel modulo, così, per assicurarvi che tutto abbia funzionato correttamente.

Modulo 2.2.3: Preparazione dello schermo

Il modulo attiva il modo grafico con i relativi colori e disegna un cerchio di 80*80 al centro dello schermo assieme al nome del diagramma.

Modulo 2.2.3: Linee 2000-2070

```
2000 REM*******************************
2010 REM DISEGNA LO SFONDO
2020 REM*******************************
2030 GRAPHIC 3,1
2040 COLOR 0,1 : COLOR 1,2 : COLOR 2,4,4 :
COLOR 3,6,4
2050 CHAR ,20-LEN(NOME$)/2,0,NOME$
2060 CIRCLE ,80,100,40,80
2070 RETURN
```

Commento

Linea 2030: Il modo grafico che stiamo per usare è un modo a colori, che ci permette di adottare tre colori diversi più quello di sfondo contemporaneamente. Il vantaggio di questa possibilità è di poter creare un diagramma facile da leggere. Esiste però un aspetto negativo. Più colori si possono avere sullo schermo contemporaneamente, più complesso e difficile è per il C16 registare il colore di ogni singolo pixel e più memoria è necessaria. Poiché lo schermo grafico si mangia da solo un bel po' di memoria, quando passiamo nel modo a colori il sistema operativo deve cercare un compromesso fra colore e dati. Una volta attivato il modo tre, la più piccola unità con cui potremo fare realmente qualsiasi cosa sarà lunga un pixel e larga due. Lo schermo, per quanto riguarda il C16, non sarà più 320 pixel in orizzontale, ma 160, consentendo così un risparmio della memoria che potrà essere utilizzata per conservare il maggior numero di colori da gestire.
Da qui in avanti, quindi, ogni volta che guardate la coordinata X (orizzontale) di qualsiasi cosa contenuta in questo programma, ricordate che dovete raddoppiarla, prima di poterla mettere a confronto con la stessa coordinata dei programmi che utilizzano il modo ad alta risoluzione.

Linea 2040: La piccola espressione dell'istruzione CHAR serve semplicemente a garantire che, indipendentemente dalla lunghezza assegnata al diagramma, il titolo risulti sempre al centro della linea in alto.

Linea 2060: Una dimostrazione di quanto abbiamo detto approposito delle dimensioni nel modo a colori. Perché la posizione e le dimensioni di questo cerchio abbiano un senso, moltiplicate il primo e il terzo parametro per due.

Verifica

Inserite queste tre linee e poi eseguite il programma:

```
1060 GOSUB 2000
1080 GETKEY A$
1090 GRAPHICO 0
```

Come risultato, otterrete la comparsa del titolo attribuito al diagramma e un cerchio bianco. Premete un tasto qualsiasi diverso da RUN/STOP e tornate allo schermo normale.

Modulo 2.2.4: Introduzione dei dettagli

Questo modulo disegna i segmenti in cui dovrà essere suddiviso il diagramma, li colora e vi abbina le etichette specificate nel modulo DATA. Per poter capire che cosa succederà, è importante che vi ricordiate bene di quanto abbiamo detto nel programma dell'orologio analogico approposito delle operazioni matematiche sul cerchio. Se ve ne siete dimenticati, tornate a quel programma e rileggetene le spiegazioni e i commenti.

Modulo 2.2.4: Linee 3000-3200

```
3000 REM*********************************
3010 REM INSERIMENTO SEGMENTI
3020 REM*********************************
3030 FOR I=0 TO MEMBRI-1
3040 R=A(I)/180*π
3050 DRAW ,80,100 TO
80+40*COS(R),100+80*SIN(R)
3060 NEXT I
3070 FOR I=0 TO MEMBRI-1
3080 R=(A(I)+4)/180*π
3090 CC=I-3*INT(I/3)+1 : IF I=MEMBRI-1 THEN
CC=0
3100 PAINT CC,80+32*COS(R),100+72*SIN(R),1
3110 TA=(A(I)+A(I+1+MEMBRI*(I=MEMBRI)))/2
3120 IF A(I+1)<A(I) THEN TA=TA+180
3130 R=TA/180*π
3140 TX=80+40*COS(R)
3150 TY=100+80*SIN(R)
3160 DX=ABS(159*(TX>80))
3170 DRAW ,TX,TY TO DX,TY
3180 CHAR
1,DX/4+(LEN(NOME$(I))-1)*(DX=159),TY/8,NOME$(I
)
3190 NEXT I
3200 RETURN
```

Commento

Linee 3030-3060: Le istruzioni permettono di disegnare una serie di linee dal centro del cerchio alla sua circonferenza, dividendolo così in segmenti. I valori usati sono quelli calcolati nel modulo 2.2.2.

Linee 3080-3100: Vengono di nuovo calcolati gli angoli dei segmenti, ma questa volta vengono aggiunti quattro gradi a ciascuno per annullare le linee disegnate dal ciclo precedente. In base ad ogni angolo viene quindi calcolata una posizione all'interno della circonferenza del cerchio. Interviene poi il comando PAINT che colora la fetta su cui è in corso l'esecuzione. La linea 3090 produce un ciclo dei tre colori del diagramma specificati nel modulo 2.2.3, eccetto quelli per l'ultimo segmento, che viene lasciato del colore di fondo. Questo per essere sicuri che l'ultimo segmento non sia dello stesso colore del primo. Poiché sono vicini l'uno all'altro, il diagramma risulterebbe difficile da leggere.

Linee 3110-3120: Queste due linee calcolano un angolo a metà fra il punto iniziale e quello finale del segmento corrente. Può accadere che nello spostamento dal primo al secondo, venga attraversata la demarcazione fra 360 e 0 del cerchio e si ottenga una cifra assurda, ma basta aggiungere 180 e l'errore è subito eliminato.

Linee 3130-3150: I valori riportati sono quelli necessari per definire un punto in relazione al raggio.
vl motivo per cui, in questa occasione, i valori SIN e COS vengono memorizzati nelle variabili TX e TY, è che sono destinati ad essere usati più volte nella stessa linea e non ci sarebbe spazio sufficiente per ripeterle ogni volta.

Linea 3160: Certamente riconoscerete l'uso di una condizione logica. Il suo effetto qui è di rendere DX uguale a 0 o a 159 a seconda del fatto che il punto definito da TX e TY sia a sinistra o a destra del centro del cerchio.

Linee 3170-3180: Qui viene tracciata una linea che va dalla circonferenza del cerchio al bordo dello schermo a sinistra o a destra in base alla definizione di DX. Al termine della linea, o meglio, sopra di essa, viene generata l'etichetta per il segmento verso il quale questa linea deve puntare. La posizione delle etichette sul lato destro viene spostata a sinistra, in modo che non vadano a finire fuori dal bordo dello schermo e ancora una volta usando una condizione logica.

Verifica

Eseguite il programma dopo aver inserito questa linea:

```
1070 GOSUB 3000
```

Vedrete comparire un'immagine simile a quella riportata dopo il titolo del paragrafo dedicato a questo secondo programma. L'unica differenza è che sul vostro schermo sarà a colori invece che in bianco e nero.

Modulo 2.2.5: Il modulo di controllo

La maggior parte delle linee di questo modulo sono già state introdutte, ma controllate comunque che siano tutte quelle elencate qui sotto, altrimenti vi mancheranno certe rifiniture piuttosto importanti.

Modulo 2.2.5: Linee 1000-1130

```
1000 REM*********************************
1010 REM MODULO DI CONTROLLO
1020 REM*********************************
1030 TRAP 1120
1040 GOSUB 5000
1050 TRAP 1030
1060 GOSUB 2000
```

```
1070 GOSUB 3000
1080 GETKEY A$
1090 GRAPHIC 0
1100 SCNCLR : LIST 4000-4999
1110 END
1120 PRINT "FORSE HAI INSERITO I DATI MALE"
1130 FOR I=0 TO 3000 : NEXT : RESUME 1100
```

Commento

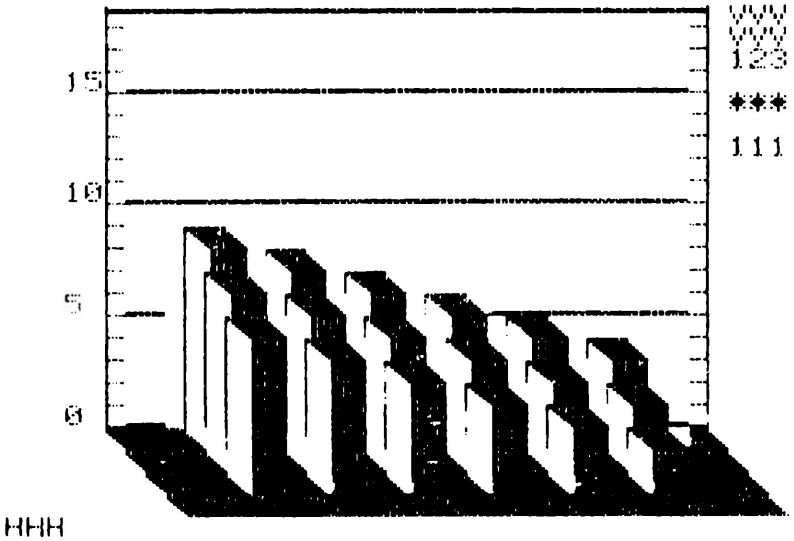
Linee 1030 e 1120-1130: Un breve avvertimento per indicare che probabilmente c'è un errore nella disposizione del modulo DATA. Questa comunque non è una prova infallibile, dato che ci sono sbagli che possono generare errori non individuati da TRAP.

Verifica

Sostituite con una lettera uno dei valori elencati sotto il titolo QUANTITÀ e poi eseguite il programma. Dovreste vedere comparire il messaggio di errore del programma e il listato del modulo DATA. Correggete lo sbaglio inserito deliberatamente e rieseguite il programma. Questa volta, premendo RUN/STOP (o un tasto qualsiasi dopo che il diagramma è stato completato) dovreste ottenere il listato del modulo DATA.

Programma 2.3: GRAPH II (Grafico tridimensionale)

Figura 2.3: Grafico tridimensionale



Scopi del programma
Dopo aver esaminato due modi diversi per presentare i dati ad alta risoluzione, vale la pena fare qualche accenno alla grande flessibilità della grafica a bassa risoluzione, in cui i computer Commodore eccellono. L'utilizzo di caratteri grafici a bassa risoluzione non solo offre all'utente una serie di effetti pronti che sarebbe molto difficile ottenere in un ambito ad alta risoluzione, ma rende possibile l'uso di una vasta gamma di colori e, cosa non trascurabile, un certo risparmio di memoria che verrebbe invece usata in modo massiccio dallo schermo ad alta risoluzione.
Nel programma che segue creeremo un diagramma a barre tridimensionali la cui riproduzione su schermo è, credo, una delle migliori dimostrazioni di quanto impressionante sia la grafica a bassa risoluzione del C16: insomma, niente da stupirvi se vi verrà voglia di chiamare tutta la famiglia per mostrare quanto siete bravi!

Modulo 2.3.1: Inizializzazione

Un modulo semplice e lineare per dichiarare un numero piuttosto limitato di variabili e vettori.

Modulo 2.3.1: Linee 2000-2090

```
2000 REM*********************************
2010 REM INIZIALIZZAZIONE
2020 REM*********************************
2030 COLOR 0,11 : COLOR 4,10 : SCNCLR
2040 CHAR,13,1,"[RVS ON][RED]GRAFICO 3D" :
PRINT
2050 DIM HH(2,6)
2060 R$=CHR$(13)
2070 CO$="[YEL][GRN][PUR]"
2080 INPUT "[BLK][CD]DEVI CARICARE I DATI DAL
DISCO (S/N):";Q$
2090 RETURN
```

Commento

Linea 2050: Il vettore HH verrà usato per memorizzare i dati del grafico. Poiché i numeri dell'istruzione DIM devono essere contati a partire da zero, ciò che viene fornito qui è lo spazio per le tre serie di sei valori.

Linea 2060: La stringa R$ verrà usata nel modulo del file dati e ne parleremo quindi in quell'occasione.

Linea 2070: La stringa è costituita da tre codici di controllo colore, cioè caratteri che, se riprodotti su schermo, modificano il colore. I tre colori nell'ordine sono il

giallo, il verde e il porpora. In seguito la stringa verrà usata per facilitare il passaggio da un colore all'altro.

Linea 2080: Quest'istruzione INPUT ha lo scopo di permettere ad un modulo, di cui parleremo più avanti, di richiamare una serie di dati su disco (o su nastro) da utilizzare per i grafici.

Modulo 2.3.2: Acquisizione dei dati

Poiché stiamo lavorando con la bassa risoluzione non abbiamo alcun bisogno di adottare certi trucchi come quello di usare le istruzioni DATA per memorizzare i cambiamenti apportati alle nostre informazioni. La maggior parte dei programmi lavorano su informazioni utili che possono essere introdotte dall'utente mentre il programma è in corso di esecuzione (stiamo parlando dei ben noti "programmi interattivi". Nel caso di questo programma, tutte le informazioni possono essere riunite insieme, in modo da avere un modulo che chiede informazioni, le usa per chiederne delle altre ed esegue poi dei controlli per verificare la correttezza dei dati ricevuti.

Modulo 2.3.2: Linee 3000-3240

```
3000 REM*********************************
3010 REM ACCETTAZIONE DEI DATI
3020 REM*********************************
3030 SCNCLR
3040 CHAR,13,1,"[RVS ON][RED]GRAFICO 3D" :
PRINT
3050 PRINT "[CD][BLK]CI SONO 19 SEZIONI
VERTICALI"
3060 INPUT "[CD][BLU]QUANTITA' RAPPRESENTATE
DA OGNI SEZIONE:";UV
3070 INPUT "[CD]COLONNE (1-6):";ND
3080 INPUT "[CD]BLOCCHI (1-3):";NB
3090 PRINT"[CD]
************************************* "
3100 INPUT "[CD][RED]NOME DELL'ASSE
ORIZZONTALE:";NH$
3110 FOR I=0 TO NB-1
3120 PRINT "[GRN]NOME DELL'ASSE
VERTICALE:";I+1;:INPUT NV$(I)
3130 NEXT I
3140 SCNCLR
3150 FOR I=0 TO NB-1
3160 FOR J=1 TO ND
```

```
3165 T=20*UV : DO UNTIL T/UV<=19
3170 PRINT "[CD]VALORE BLOCCO";I+1"
COLONNA";J;":";
3180 INPUT T
3200 IF INT(T/UV)>19 THEN PRINT "[CD]VALORE
TROPPO ALTO."
3210 LOOP
3220 HH(I,J)=T
3230 NEXT J,I
3240 RETURN
```

Commento

Linee 3050-3060: Come nel caso del primo grafico, ogni singola unità dell'asse verticale può rappresentare qualsiasi valore specificato dall'utente, ma poiché stiamo lavorando in ambiente a bassa risoluzione, non disponiamo della stessa flessibilità per quanto concerne le dimensioni delle unità verticali. Le sole dimensioni di una certa utilità per le singole unità è l'altezza di un quadro carattere che ci permette di avere, per il formato del nostro grafico, 19 unità sull'asse verticale.

Linee 3070-3080: Il grafico ci permetterà di presentare, come abbiamo già detto, tre serie di sei valori ciascuna, che verranno visualizzate come tre file di sei colonne di barre.

Linee 3110-3130; Poiché il diagramma può contenere fino a tre file, devono essere previsti tre nomi per l'asse orizzontale (ad esempio, prezzo di vendita, costo, guadagno). Notate che i nomi devono essere inseriti nel vettore NV$ (Nome per Verticale) a partire dall'elemento zero. Questo vettore non viene dimensionato nella routine di inizializzazione perché avrà solo e sempre tre elementi. È sufficiente la semplice menzione del nome del vettore nel programma perché al vettore vengano assegnati dieci elementi (0-9).

Linee 3150-3230; Qui viene chiesto all'utente di introdurre per ogni fila i dati relativi al numero di colonne specificato. Appena introdotto, il numero viene controllato per verificare che non sposti il grafico oltre il segno corrispondente alla diciannovesima unità. Lo scopo del ciclo alle linee 3165-3210 è quello di garantire che non vengano introdotti numeri che richiederebbero un grafico più alto dei diciannove quadri-carattere dello schermo.

Verifica

Battete queste tre linee (in seguito alcune di loro diventeranno parte del modulo di controllo del programma):

```
1030 GOSUB 2000
1040 GOSUB 3000
1090 SCNCLR: END
```

Ora eseguite il programma che avete introdotto finora e rispondete alle domande nel modo seguente:
Per il numero rappresentato da ciascuna unità verticale: 1
Colonne: 1
File; 1
Nome dell'asse orizzontale: ORIZZONTALE
Nome dell'asse verticale 1: VERTICALE 1
File 1, valore 1: 10
Dopo aver introdotto l'ultimo numero, lo schermo viene azzerato e il programma genera il messaggio READY fermandosi subito dopo. Battete:
?,UV,ND,NB,NH$,NV$(0),HH(0,1)

e vedrete comparire la scritta:

1 1 1 ORIZZONTALE VERTICALE 1 10

Se volete, potete eseguire di nuovo il programma e provare a battere dei valori non validi. Non dovreste più essere in grado di introdurre nessun valore relativo alla colonna che sia superiore a 19 volte il valore della singola unità dell'asse verticale.

Modulo 2.3.3: Disegno della griglia

Come i grafici linerari, anche questo ha bisogno di una griglia che permetta una rappresentazione dei dati con un preciso significato. La griglia è realizzata da questo modulo che colloca i caratteri a bassa risoluzione nella giusta posizione sullo schermo usando variabili di ciclo e il comando CHAR.

Modulo 2.3.3: Linee 4000-4240

```
4000 REM*********************************
4010 REM DISEGNA LO SFONDO
4020 REM*********************************
4030 COLOR 0,1
4040 SCNCLR
4050 FOR I=0 TO 3
4060 CHAR,5+I,20+I,"[BRN]";CHR$(127);"[RVS ON]
                                  ";CHR$(127);"[RVS
OFF]"
4070 NEXT
4080
CHAR,5,0,"[WHT]_____________________________________"
4090 FOR I=1 TO 19
4100 CHAR,5,I,CHR$(108)
4110 CHAR,34,I,CHR$(186)
```

```
4120 NEXT
4130 FOR I=4 TO 19 STEP 5
4140 CHAR,6,I,"____________________________"
4150 NEXT
4160 CHAR,0,24,NH$
4170 FOR H=0 TO NB-1
4180 PRINT MID$(CO$,H+1,1);
4190 TT$=NV$(H)+" *"+STR$(UV)
4200 FOR I=1 TO LEN(TT$)
4210 CHAR,36+H,I,MID$(TT$,I,1)
4220 NEXT I,H
4230 PRINT
"[HOME][CR][CR][CR][CD][CD][CD][CD]15[CL][CL][
CD][CD][CD][CD][CD]10[CL][CL][CD][CD][CD][CD][
CD]5[CL][CD][CD][CD][CD][CD]0"
4240 RETURN
```

Commento

Linee 4050-4070: Questo ciclo riproduce una linea di trenta spazi marroni in negativo con cui fornisce la base sulla quale verrà realizzato il grafico. Il carattere grafico è ottenuto con il tasto *.

Linea 4080: Una linea trasversale in alto sullo schermo, costituita dal carattere grafico sopra la P battuto 30 volte.

Linee 4100-4110: Le linee verticali su ciascuna estremità del grafico, che contraddistingono ogni singola unità. I caratteri grafici sono sui tasti L e P.

Linee 4130-4150: Quattro linee orizzontali che suddividono le unità in gruppi di cinque lungo l'asse verticale.

Linea 4160: L'etichetta dell'asse orizzontale.

Linee 4170-4220: Questi cicli riproducono il o i nomi per l'asse verticale lungo il lato destro dello schermo, assieme al valore che ciascuna unità rappresenta. Notate l'uso di MID$ per ricavare un solo carattere da CO$ e per riprodurlo sullo schermo, cambiando in questo modo il colore di ciò che viene visualizzato. Il colore di ciascun titolo corrisponde al colore di una delle file. L'uso di CO$ in questo modo permette di specificare facilmente qualsiasi sequenza di colori (basta modificare i caratteri di controllo nella stringa).

Linea 4230: Questa linea contraddistingue l'asse verticale sinistro usando solo i caratteri di controllo del cursore invece di un ciclo e CHAR.

Verifica

Battete questa linea:

```
1050 GOSUB 4000
```

ed eseguite il programma. Specificate il valore dell'unità, una colonna e tre file. Assegnate tre nomi all'asse verticale e quando vi viene chiesto di specificare il valore delle colonne, premete semplicemente il tasto RETURN. Dovreste veder comparire la griglia di appoggio per il diagramma, con il nome dell'asse orizzontale in basso e i nomi di quello verticale in alto a destra.

Modulo 2.3.4: Traccia del grafico

Questo modulo sarà, a dire il vero, più facile da capire dopo essere stato introdotto e dopo che se ne saranno visti i risultati. Esso non è basato tanto su principi di fondo quanto sui risultati degli esperimenti fatti per capire quali combinazioni di caratteri e quali posizioni di stampa producevano i risultati desiderati.

Modulo 2.3.4: Linee 5000-5220

```
5000 REM*******************************
5010 REM DISEGNA I BLOCCHI
5020 REM*******************************
5030 FOR H=0 TO NB-1
5040 PRINT MID$(CO$,H+1,1)
5050 FOR I=ND TO 1 STEP-1
5060 CHAR,8+4*(I-1)+H,20+H,""
5070 DO WHILE INT(HH(H,I)/UV)<>0
5080 FOR J=1 TO INT(HH(H,I)/UV)
5090 IF J=1 THEN PRINT "[BRN][RVS ON]
";CHR$(127);MID$(CO$,H+1,1);"| [RVS
OFF][CU][CL][CL][CL][CL]";
5100 IF J>1 THEN PRINT " |[RVS ON]| [RVS
OFF][CU][CL][CL][CL][CL]";
5110 NEXT J
5120 PRINT " |";CHR$(127);"[RVS ON] ";CHR$(127)
5130 EXIT : LOOP
5140 NEXT I
5150 NEXT H
5160 FOR I=0 TO ND-1
5170 CHAR,3+4*I,20,""
5180 FOR J=1 TO NB
5190 IF J>1 OR HH(2,I)=0 OR (J=1 AND ND=0)
```

```
THEN PRINT "[BRN][RVS ON]";CHR$(127);"[CL]";
5200 PRINT "[CD][CR]";
5210 NEXT J,I
5220 RETURN
```

Commento

Linne 5030 e 5150: Questo ciclo crea il numero di file specificato.

Linee 5050 e 5140: Il secondo ciclo crea il numero di colonne.

Linea 5060: La posizione di partenza per la traccia di ogni singola colonna. Essa si sposterà di quattro spazi a sinistra per ogni nuova colonna a partire dall'estremità destra. Appena completata una fila, verrà riprodotta quella successiva uno spazio in giù e a destra rispetto alla precedente.

Linee 5070 e 5130: Queste linee creano un ciclo fittizio per la parte che disegna una colonna se il valore per quest'ultima è uguale a zero. Il ciclo è detto fittizio in quanto non viene eseguito mai più di una volta e il concetto di ciclicità viene quindi meno.

Linee 5080-5110: Il ciclo che disegna la parte principale della colonna a tre dimensioni. La variabile del ciclo viene impostata in modo da cambiare da uno fino all'altezza della colonna. Al primo passaggio nel ciclo, viene creata la base della colonna (per passare dal colore marrone della base al colore della fila che viene disegnata in quel momento. Nei passaggi successivi, utilizzando diversi caratteri, il ciclo crea le facce laterali e quella frontale della colonna. Notate che il movimento verso l'alto del disegno viene ottenuto usando i caratteri di controllo del cursore e non CHAR. Usare quest'ultima significa dover sapere sempre la propria posizione sullo schermo, mentre, usando i caratteri di controllo del cursore, è sufficiente sapere dove inizia la colonna e poi muoversi di conseguenza fino a completarla.

Linea 5120: Viene aggiunto il dato relativo al bordo superiore della colonna.

Linee 5160-5190: Terminate le colonne rimangono in basso i loro bordi mal definiti: queste linee servono a ripulirle.

Verifica

Battete questa linea:

```
1060 GOSUB 5000
```

ed eseguite il programma. Indicate uno come valore dell'unità e specificate tre colonne e tre file. I nomi per gli assi sono irrilevanti perciò sceglieteli come vi pare. Quando il programma vi chiede i valori delle colonne battete:

```
6,12,18,6,12,18,6,12,18
```

A questo punto dovreste veder comparire sullo schermo tre file e tre colonne, con le facce superiori di queste ultime che formano una superficie liscia a partire dalla fila dietro a quella davanti. Notate che nel leggere i valori per le tre file, dovete partire dal presupposto che la faccia superiore della barra più in avanti continua verso il fondo e fino al punto più arretrato. Nell'esempio sullo schermo, quello che avete davanti a voi sono tre colonne, in ciascuna delle quali le tre barre rappresentano lo stesso valore, anche se quella davanti è fisicamente la più bassa. Questo è reso necessario dal fatto di dover conservare l'illusione della tridimensionalità.
Fate i vostri esperimenti con il programma e osservate in che modo esso tratta i diversi dati introdotti. Scoprirete che funzionerà bene solo con dati per i quali ogni fila non è mai più alta di quella che le sta' dietro. Questo tipo di grafico si adatta a varie situazioni reali: può andare benissimo, ad esempio, per rappresentare il rapporto fra prezzo di vendita, costo e guadagno in una relazione sulle vendite di un'azienda.

Modulo 2.3.5: Memorizzazione dei dati su disco o su nastro

Come nel caso del programma DAYPLAN, analizzato nel primo capitolo, questo programma prevede la conservazione dei dati su disco o su nastro. Per maggiori informazioni sul metodo seguito, vi consigliamo di ripassare i commenti riservati a quel programma.

Modulo 2.3.5: Linee 6000-6190

```
6000 REM*********************************
6010 REM DATI SU FILE
6020 REM*********************************
6030 OPEN 1,8,2,"@0:GRAPHDATA,S,W"
6040 PRINT#1,NB;R$;ND;R$;NH$;R$;UV
6050 FOR I=0 TO NB-1
6060 PRINT#1,NV$(I)
6070 FOR J=0 TO ND
6080 PRINT#1,HH(I,J)
6090 NEXT J,I
6100 PRINT#1 : CLOSE1
6110 RETURN
6120 OPEN 1,8,2,"GRAPHDATA,S,R"
6130 INPUT#1,NB,ND,NH$,UV
6140 FOR I=0 TO NB-1
6150 INPUT#1,NV$(I)
6160 FOR J=0 TO ND
6170 INPUT#1,HH(I,J)
```

```
6180 NEXT J,I
6190 CLOSE1 : RETURN
```

Commento

Linea 6040: Una cosa da notare in questo modulo, approposito dell'istruzione PRINT$, è la presenza sulla linea di un certo numero di R$. Ricorderete che nel primo modulo del programma, R$ era posta uguale a CHR$ (13), cioé il carattere RETURN, che sta' a significare la fine del dato da trasferire. Quando si devono trasferire in un file più voci usando la stessa istruzione PRINT$, tutte le voci verranno elaborate insieme, a menoché tra esse non venga posto un R$. Le variabili effettivamente trasferite nel file sono semplicemente alcuni dei valori essenziali che sono stati introdotti nel secondo modulo del programma.

Linea 6050: Nel commento all'ultima linea, abbiamo detto che R$ (o qualsiasi altra variabile uguale a CHR$(13)), viene inserita nella linea per separare le voci l'una dall'altra. Ma allora, perché la stessa cosa non è stata fatta anche per questi due cicli che trasferiscono il contenuto di due vettori nel file su disco? La risposta è che ogni volta che un'istruzione PRINT o PRINT$ termina senza segni di punteggiatura, il C16 fa' seguire automaticamente l'ultima voce da un RETURN (ecco perché, se una certa voce non ha una virgola o un punto e virgola in fondo, fa' si che quella che la segue venga riprodotta sullo schermo su un'altra linea).

Linee 6080-6110: Come potete notare, non abbiamo usato il separatore R$ durante l'introduzione dei dati con l'istruzione INPUT. È tipico delle istruzioni INPUT e INPUT$ non saper riconoscere di aver ricevuto un valore fintantoché non viene premuto o non viene letto sul disco il tasto RETURN.

Verifica

La verifica di questo modulo è significativa solo dopo l'introduzione delle poche linee riportate qui di seguito e che completano il modulo di controllo del programma.

Modulo 2.3.6: Linee 1000-1110

```
1000 REM*********************************
1010 REM MODULO DI CONTROLLO
1020 REM*********************************
1030 GOSUB 2000
1040 IF Q$="S" THEN GOSUB 6120 : ELSE GOSUB
3000
1050 GOSUB 4000
1060 GOSUB 5000
1070 GETKEY A$
1080 CHAR ,0,24,""
```

```
1090 INPUT "EREGISTRI I DATI (S/N):";Q$
1100 IF Q$="S" THEN GOSUB 6000
1110 SCNCLR : END
```

Verifica

Eseguite semplicemente il programma. A questo punto dovreste poter introdurre i dati per la realizzazione del grafico. Non appena il grafico è stato visualizzato, premete un tasto qualsiasi e il programma vi chiederà se volete salvare i dati. Se rispondete con Y, eseguite di nuovo il programma e, quando vi verrà chiesto se volete caricare i dati da disco, rispondete ancora con Y. Dovreste vedere sullo schermo esattamente lo stesso grafico.

CAPITOLO 3
SUONI E LUCI

In questo capitolo daremo un'occhiata da vicino alle capacità del C16 nel campo del suono e della grafica.
Nella maggior parte dei casi, i vostri programmi si accontenteranno di piccole routine incorporate già predisposte per l'esecuzione di processi legati alla grafica e al suono. Questo perché ciò che occorre è sempre qualcosa di molto elementare come il modulo per l'inserimento di un titolo, visto nell'ultimo capitolo, o quello per la generazione di segnali acustici a due tonalità. Vi sono però casi in cui è necessario disporre di qualcosa di più sofisticato, come un disegno complesso o un breve brano musicale per rendere il programma più professionale. In questi casi, invece di creare un disegno o un suono ogni volta che se ne ha bisogno, scrivendo una routine a parte, è molto più pratico avere degli strumenti dedicati che semplifichino la creazione dei disegni e permettano poi di richiamarli da disco o nastro in un successivo momento.
In questo capitolo imparerete ad usare quattro di questi strumenti, con i quali potrete creare vostri disegni sia a bassa che ad alta risoluzione, o serie di caratteri di vostra invenzione, e di scrivere ed editare brani musicali usando sempre e solo il vostro C16. Con le funzioni introdotte nei programmi che vi presentiamo in questo capitolo, il limite all'uso del suono e della grafica sarà unicamente la vostra immaginazione.
I programmi presentati in questo terzo capitolo sono:

ARTIST, che permette di creare disegni a bassa risoluzione

DISPLAY, che visualizza rapidamente una serie di immagini create con il programma ARTIST.

CHARACTERS, che consente di creare set di dati personalizzati.

HDRAW, uno strumento per la creazione di disegni ad alta risoluzione.

MUSIC, che consente di introdurre suoni doppi in un formato molto semplice e di ascoltarli poi riprodotti dal C16.

Programma 3.1: Artist

Scopi del programma

Nel programma relativo al grafico tridimensionale presentato nel secondo capitolo, abbiamo visto che il set di caratteri a bassa risoluzione del C16 è in grado di creare effetti davvero sorprendenti, effetti che, fra l'altro, possono essere ottenuti

usando non più della metà della memoria necessaria per creare immagini ad alta risoluzione e lasciando così memoria sufficiente per l'introduzione e l'esecuzione di programmi davvero utili. Lo scopo di questo programma è di utilizzare profiquamente il set di caratteri a bassa risoluzione, trasformando lo schermo in una pagina bianca sulla quale poter muovere il cursore, scrivere i caratteri, cancellarli e modificarne il colore fino ad arrivare alla versione definitiva da poter trasferire su disco o su nastro per impieghi futuri.
Nel programma che stiamo per analizzare vengono presentati per la prima volta questi nuovi concetti:

1) Inizializzazione automatica dei programmi
2) Creazione di un cursore controllato dall'utente
3) Scrittura di caratteri e colori sullo schermo (POKE)
4) Uso di comandi a un tasto senza menu
5) Funzione HELP
6) Memorizzazione dell'immagine
7) Salvataggio in un file di dati contenente i disegni su schermo
8) Generazione di messaggi di errore dopo un TRAP
9) Specificazione interattiva dei nomi dei file su disco

Modulo 3.1.1: Inizializzazione

Un'occhiata a questo modulo sarà sufficiente a convincervi che il programma che state per introdurre nel computer è più complesso di quelli proposti finora.

Modulo 3.1.1: Linee 10000-10120

```
10000 REM*******************************
10010 REM VARIABILI
10020 REM*******************************
10030 DO UNTIL IN : IN=1
10040 R$=CHR$(13) : FL=0 : RV=0
10050 C1=2 : C0=6 : L1=7 : L0=3 : PC=1 : PL=7
10060 COLOR 4,8,3 : COLOR 0,C0,L0 : COLOR
1,C1,L1
10070 MENU$="CVBNPRF[CLEAR][HOME]SDLHE"
10080 DEF FNCV(X)=PEEK(3072+40*I+J)
10090 DEF FNCC(X)=PEEK(2048+40*I+J)
10100 CL$="
    ":REM 38 SPAZI
10110 GOSUB 15000
10120 LOOP
```

Commento

Linee 10030 e 10120: Queste due linee rappresentano un modo molto semplice per ottenere una prestazione davvero utile, nota come "inizializzazione automatica". Nel nostro caso, essa sta' ad indicare che d'ora in avanti, se questo programma viene avviato mediante l'uso dell'istruzione GOTO 1 (supponendo che sia stato inserito il piccolo modulo SAVE riportato nel capitolo 1), non reinizializzerà tutte le variabili. Nei programmi che gestiscono i dati, questa capacità può essere di importanza essenziale, in quanto consente di continuare a lavorare sui dati esistenti nel caso in cui il programma sia stato interrotto per errore.
Nel caso specifico di questo programma esso non è usato con questo scopo per la semplice ragione che, se avete interrotto il programma, il disegno su schermo è già stato rovinato e non ha alcun senso conservarlo. Più avanti nel programma, però, scopriremo che una certa tecnica di ricaricamento di disegni su disco o su nastro fa' sì che il programma venga eseguito dall'inizio. Poiché il valore di IN è già stato posto a uno, le variabili non verranno reinizializzate e il disegno sullo schermo non verrà cancellato.

Linee 10040-10050: Ecco l'elenco delle principali variabili dichiarate:

R$ — separatore da usare nei file di dati
FL — indicatore dell'attivazione del lampeggiamento del cursore
RV — indicatore dell'attivazione della visualizzazione in negativo
C1, L1 — il colore e la luminosità dei caratteri da stampa
CO, LO — il colore e la luminosità dello sfondo dello schermo
PC, PL — il colore e la luminosità del cursore lampeggiante e dei solleciti su schermo

Linea 10070: Questa stringa verrà usata in seguito per produrre una serie di comandi a un tasto per il controllo dell'esecuzione del programma.

Linee 10080-10090: Queste due funzioni verranno usate per puntare ad un certo byte sullo schermo e nella memoria colori.

Linea 10100: Questa linea di spazi verrà usata per cancellare i solleciti comparsi sullo schermo.

Linea 10110: Una chiamata alla routine successiva che azzera lo schermo e colloca il cursore lampeggiante nell'angolo in alto a sinistra.

Verifica

A questo punto potete fare una piccola verifica introducendo temporaneamente la linea:

```
15000 RETURN
```

ed eseguendo il modulo. Se la sua introduzione è avvenuta correttamente, dovreste vedere lo schermo diventare verde e senza messaggi di sorta.

Modulo 3.1.2: Azzeramento dello schermo

Questo semplice modulo non ha bisogno di spiegazioni. L'unica cosa da dire è che X1 e Y1 vengono usate per registrare le coordinate rispettivamente orizzontale e verticale del cursore definito dall'utente, che sarà creato dal prossimo modulo.

Modulo 3.1.2: Linee 15000-15050

```
15000 REM*******************************
15010 REM CANCELLA LO SCHERMO
15020 REM*******************************
15030 SCNCLR
15040 X1=0 : Y1=0
15050 RETURN
```

Verifica

Battete GOTO 15000 e dovreste ottenere per risposta la cancellazione dello schermo e poi l'interruzione dell'esecuzione con la visualizzazione del messaggio RETURN WITHOUT GOSUB.

Modulo 3.1.3: Il cursore lampeggiante

Ci sono moltissime applicazioni serie e meno serie che possono risultare più efficaci se in grado di utilizzare un cursore definito in base a criteri più consoni e mosso sullo schermo sotto il controllo del programma. Ebbene con questo modulo, ottenere un cursore di questo tipo diventa estremamente facile.

Modulo 3.1.3 Linee 12000-12160

```
12000 REM*******************************
12010 REM LAMPEGGIO CURSORE
12020 REM*******************************
12030 P1=3072+40*Y1+X1
12040 P2=P1-1024
12050 CV=PEEK (P1)
12060 CC=PEEK (P2)
12070 COLOR 1,PC,PL
12080 A1$="" : DO UNTIL A1$<>""
12090 CHAR ,X1,Y1,"*"
12100 FOR I=1 TO 50 : NEXT
12110 POKE P1,CV : POKE P2,CC
12120 FOR I=1 TO 50 : NEXT
12130 GET A1$
```

```
12140 LOOP
12150 COLOR 1,C1,L1
12160 RETURN
```

Commento

Linee 12030-12040: Queste due variabili verranno usate per puntare a due locazioni della memoria del C16. P1 indica la posizione della memoria che contiene uno dei caratteri dello schermo (la memoria dello schermo inizia dal byte numero 3072). P2, invece, punta al byte di memoria che contiene i dati relativi al colore del carattere (la memoria colore inizia dal byte numero 2048).

Linee 12050-12060: I valori per il carattere (Character Value) e per il colore (Character Colour) sui quali va collocato il cursore lampeggiante (in corrispondenza della posizione X1,Y1).

Linea 12070: Il colore per la stampa è cambiato in quello per il cursore.

Linee 12080-12140: In questo ciclo di istruzioni, che continua finché non viene premuto un tasto, il cursore viene continuamente riprodotto sullo schermo nel punto X1, Y1 e poi, dopo una breve pausa, il carattere originale, il cui valore e il cui colore sono conservati in CV e CC, viene riconfermato scrivendo (con POKE) i valori originari sullo schermo e nella memoria colore. L'effetto di tutto ciò è la ricomparsa sullo schermo del carattere originale. Continuare l'elaborazione per mezzo del ciclo significa che il cursore asterisco compare lampeggiante sopra il carattere sullo schermo. Ogni volta che viene premuto un tasto, il ciclo termina con il carattere originale sullo schermo in modo che il cursore non cambi nulla passandoci sopra.

Linea 12150: Viene attivato il colore di stampa normale.

Verifica

Eseguite semplicemente il programma. Dovreste vedere un asterisco nero lampeggiante nell'angolo in alto a sinistra dello schermo. Se provate a premere un tasto qualsiasi, il programma dovrebbe fermarsi e generare il messaggio di errore RETURN WITHOUT GOSUB. Non potete ancora muovere il cursore, salvo modificando temporaneamente i valori di X1 e Y1 nella linea 15040.

Modulo 3.1.4: Un menu invisibile

Poiché l'obiettivo di questo programma è creare un disegno sullo schermo, l'ultima cosa che vogliamo è che sullo schermo compaia un menu che ci distrugga tutto il lavoro fatto. In questo programma, dobbiamo ricorrere necessariamente ad una tecnica diversa, una tecnica dove i comandi siano disponibili sotto forma di singoli tasti che l'utente deve imparare ad usare, servendosi all'inizio della funzione HELP. L'obiettivo di questo modulo è di analizzare i dati introdotti nel programma e stabilire se devono essere visualizzati o interpretati come comandi.

Modulo 3.1.4: linee 11010-11180

```
11000 REM*********************************
11010 REM CURSORE - MOVIMENTO - STAMPA
11020 REM*********************************
11030 TRAP 11180
11040 DO
11050 A1$="" : DO UNTIL A1$="£"
11060 GOSUB 12000
11070 IF A1$<>"£" THEN GOSUB 13000
11080 LOOP
11090 GOSUB 12000
11100 A2$=A1$
11110 Z=INSTR(MENU$,A1$)
11120 IF Z>0 AND Z<6 THEN Z=5
11130 Z=Z-4 : IF Z<0 THEN Z=0
11140 ON Z GOSUB
14000,16000,17000,15000,15040,18000,13000,2300
0,25000,13000
11150 COLOR 0,C0,L0 : COLOR 1,C1,L1
11160 LOOP
11180 SCNCLR : PRINT ERR$(ER),EL : END
```

Commento

Linee 11030 e 11180: Una TRAP molto lineare per porre fine all'esecuzione del programma se viene premuto il tasto STOP. Notate, comunque, che questa volta, quando il programma si ferma, compaiono sullo schermo il messaggio di errore significativo e la linea alla quale si riferisce, per effetto delle variabili di sistema ERR$, ER e EL. In questo modo, se saltano fuori degli errori, la trappola non maschera il messaggio di errore. Nella maggior parte dei casi, ciò che viene visualizzato è un BREAK accompagnato dal numero della linea in corso di esecuzione nel momento in cui è stato premuto il tasto STOP.

Linee 11000–11160: Il ciclo principale che continua a passare attraverso questo modulo di controllo.

Linee 11050-11080: Questo ciclo distribuisce i dati in ingresso fra il modulo successivo (linea 13000), che li riproduce sullo schermo, e le linee successive a questo modulo, che hanno a che fare con i comandi. Questi ultimi vengono riconosciuti perché preceduti dal tasto con la lira sterlina (o con il cancelletto) che si trova in alto a sinistra sulla tastiera. Gli altri tasti carattere normali, se premuti, producono la comparsa del carattere sullo schermo in corrispondenza

della posizione del cursore (anche se ciò avviene solo dopo che è stato introdotto il modulo alla linea 13000). Ricordate che per ottenere il segno della lira sterlina sullo schermo, dovete battere il tasto due volte.

Linea 11090: Questa linea viene raggiunta in fase di esecuzione solo se è stato premuto il tasto con la lira sterlina. La sua funzione consiste nell'invocare la funzione "cursore lampeggiante/attesa" in modo che il modulo di input riceva ancora un comando ad un tasto.

Linea 11100: Il valore corrente di A1$ viene memorizzato in A2$ dato che può essere modificato da un modulo successivo.

Linee 11110-11140: Ci siamo già trovati di fronte a ON... GOSUB e abbiamo visto che rappresenta un modo per saltare alle varie parti del programma. La novità qui è l'uso del comando INSTR per trovare il valore della variabile Z. Tutto ciò che il comando fa' è cercare in tutta la stringa MENU$, che è stata definita alla linea 10070 del modulo 3.1.1., e comunicare se il tasto che è stato premuto è presente e in quale punto. Se non è inserito nella stringa, il valore fornito da INSTR è zero e ON... GOSUB viene aggirata. In altri termini, INSTR è un modo molto semplice di prendere un comando sotto forma di carattere e di trasformarlo in un numero a seconda della posizione del carattere nella stringa MENU$. Le tre possibilità di trattamento del valore di Z devono tener conto del fatto che i primi cinque comandi a un tasto in MENU$ devono essere trattati con un'unica subroutine. Se il carattere introdotto è uno dei primi cinque, il valore di Z è 5, da cui si toglie quattro in modo che Z abbia un valore da uno a dieci.

Verifica

Poiché mancano ancora le subroutine più importanti, il modulo non può essere ancora sottoposto a verifica. Se comunque introduciamo temporaneamente la linea:

```
13000 RETURN
```

dovremmo scoprire che, se eseguiamo il programma, comparirà il cursore lampeggiante. In questa situazione, la pressione di un tasto carattere qualsiasi, compresi i caratteri grafici, non produrrà alcun effetto. Ora provate a premere invece il tasto con la lira sterlina seguito da una delle lettere contenute nella stringa MENU$. Il programma dovrebbe fermarsi visualizzando il messaggio di errore UNDEFINED LINE. Impartite di nuovo il comando RUN e questa volta premete STOP non appena compare il cursore lampeggiante. Lo schermo si azzererà e in alto a sinistra comparirà la parola BREAK.

Modulo 3.1.5: Spostamento del cursore e visualizzazione

Questo è l'ultimo dei moduli che costituiscono l'ossatura del programma. I prossimi moduli saranno solo una rifinitura. Il modulo è costituito da una routine che

acquisisce i dati in ingresso diversi dai comandi e li visualizza o li usa per muovere il cursore sullo schermo.

Modulo 3.1.5: Linee 13000-13130

```
13000 REM*********************************
13010 REM STAMPA NORMALE
13020 REM*********************************
13030 IF X1>0 AND A1$="[CL]" THEN X1=X1-1 :
RETURN
13040 IF X1<38 AND A1$="[CR]" THEN X1=X1+1 :
RETURN
13050 IF Y1>0 AND A1$="[CU]" THEN Y1=Y1-1 :
RETURN
13060 IF Y1<23 AND A1$="[CD]" THEN Y1=Y1+1:
RETURN
13070 IF A1$=CHR$(20) THEN CHAR ,X1,Y1," "
13080 IF A1$<"!" OR (A1$>CHR$(127) AND
A1$<CHR$(161)) THEN RETURN
13090 IF RV THEN A1$="[RVS ON]"+A1$
13100 IF FL THEN A1$="[FLASH ON]"+A1$
13110 CHAR ,X1,Y1,A1$
13120 PRINT "[RVS OFF][FLASH OFF]";
13130 RETURN
```

Commento

Linee 13030-13060: Queste linee modificano il valore di X1 o Y1, le variabili relative alla posizione del cursore, se viene usato uno dei tasti con freccia. Le condizioni inserite nelle linee stanno ad indicare che il cursore non può spostarsi né oltre il bordo dello schermo, né sulle due linee più in basso, che sono riservate ai solleciti generati dal programma.

Linea 13070: Premendo il tasto DEL, nella posizione corrente viene generato uno spazio che cancella quello precedente.

Linea 13090: Il modulo accetta solo i normali caratteri di stampa (vedi tabella dei caratteri ASCII nel manuale).

Linee 13090-13100: Se l'utente ha selezionato la visualizzazione in negativo o il lampeggiamento, alla stringa di caratteri da visualizzare viene aggiunto un carattere RVS ON o FLASH ON.

Linee 13110-13120: Il carattere viene stampato nel punto X1, Y1. Dato che CHAR è uguale all'istruzione PRINT con l'aggiunta di un punto e virgola, la

visualizzazione in negativo o il lampeggiamento potrebbero rimanere attivati su tutto quello che viene stampato da qui in avanti. FLASH OFF e RVS OFF hanno qui lo scopo di annullare le due caratteristiche.

Verifica

A questo punto dovreste essere in grado di muovere il cursore sullo schermo e di riprodurvi (con l'esclusione delle due linee più in basso), qualsiasi lettera, numero o carattere grafico. Ricordate che per ora non avete alcun controllo né sul colore in cui devono comparire i caratteri, né sul lampeggiamento o la visualizzazione in negativo.

Modulo 3.1.6: La funzione HELP

Quando un programma non presenta immediatamente le proprie funzioni tramite menu, può creare nell'utente una certa forma di frustrazione. Ma esistono certi programmi, come gli word processor, che non potrebbero mai elencare in una volta sola le loro centinaia di comandi disponibili durante l'esecuzione.
Anche nel caso di questo programma, comunque, ci sono ben 10 scelte possibili per i comandi, nessuna delle quali è presentata in un menu. La soluzione, in questo caso, è data da quella che è ormai nota come funzione "HELP" (di pronto soccorso) e permette all'utente di chiedere informazioni durante la fase di esecuzione del programma. Il modulo che stiamo per esaminare, propone un elenco di brevi messaggi che hanno la sola funzione di ricordare all'utente i diversi comandi che può utilizzare e i tasti da usare per impartirli.

Modulo 3.1.6: Linee 25000-25260

```
25000 REM*******************************
25010 REM HELP
25020 REM*******************************
25030 DATA *=SOSPENDE HELP
25040 DATA C=COLORE TESTO RIGA SUPERIORE
25050 DATA V=COLORE TESTO RIGA INFERIORE
25060 DATA B=COLORE SFONDO RIGA SUPERIORE
25070 DATA N=COLORE SFONDO RIGA INFERIORE
25080 DATA P=COLORE CURSORE (SOLO RIGA SUP.)
25090 DATA R=INTERRUTTORE REVERSE
25100 DATA F=INTERRUTTORE FLASH
25110 DATA HOME=CURSORE INIZIO VIDEO
25120 DATA CLEAR=CANCELLA IL VIDEO
25130 SATA S=SALVA IL DISEGNO
25140 DATA D=DEFINISCE GLI ANGOLI E SALVA-->
25150 DATA IL DISEGNO QUANDO IL SECONDO -->
```

```
25160 DATA ANGOLO E' STATO DEFINITO
25170 DATA L=CARICA DISEGNO GIA' SALVATO
25180 DATA FINE
25190 RESTORE 25000 : TT$=""
25200 COLOR 1,PC,PL : DO UNTIL TT$="*"
25210 READ HH$ : IF HH$="FINE" THEN EXIT
25220 CHAR ,0,24,HH$
25230 GETKEY TT$
25240 GOSUB 24050
25250 LOOP : COLOR 1,C1,L1
25260 RETURN
```

Commento

Linee 25030-25180: Questi sono i messaggi che verranno visualizzati.

Linee 25200-25250: Questo ciclo stampa i singoli messaggi contenuti nelle instruzioni DATA fino a quando è raggiunta la parola END o viene premuto il tasto "*" (asterisco).

Verifica

Rimandate le prove a dopo aver introdotto il prossimo modulo, che cancella qualsiasi messaggio visualizzato nell'ultima linea in basso sullo schermo.

Modulo 3.1.7: Cancellazione della linea dei messaggi

Tutti i messaggi diretti dal programma all'utente vengono riportati nelle ultime due linee dello schermo. Questo modulo ha il compito di cancellare tutti i messaggi visualizzati su queste linee. In base alla linea da cui viene chiamato, può creare una pausa e poi cancellare tutte e due le linee, cancellare semplicemente le due linee oppure cancellare solo l'ultima delle due.

Modulo 3.1.7: Linee 24000-24060

```
24000 REM*******************************
24010 REM CANCELLA 23 E 24 LINEA
24020 REM*******************************
24030 FOR I=1 TO 2000 : NEXT
24040 CHAR ,0,23,CL$
24050 CHAR ,0,24,CL$
24060 RETURN
```

Verifica

Adesso potete andare avanti con la verifica e controllare questo modulo e quello precedente.

Impartite il comando RUN, per eseguire il programma, e poi introducete il tasto con la lira sterlina, seguito dal tasto H. I due tasti dovrebbero provocare la comparsa del messaggio:

SOSPENDE HELP

in fondo allo schermo. Premendo la barra spaziatrice, potrete scorrere lungo tutti i messaggi generati da HELP. Per interrompere la funzione, sarà sufficiente battere il tasto "*".

Modulo 3.1.8: Impostazione della visualizzazione in negativo e del lampeggiamento

Riprodurre semplicemente dei caratteri sullo schermo non basta quando si vogliono creare dei programmi grafici di aspetto quasi professionale. Per fortuna, il C16 ci permette di scegliere se riprodurli in un certo colore, visualizzati o meno in negativo ed eventuawmente lampeggianti. Questo modulo, o per meglio dire, questi due brevi moduli hanno per l'appunto lo scopo di permettere la scelta del negativo e del lampeggiamento.

Modulo 3.1.8: Linee 16000-17030

```
16000 REM*******************************
16010 REM INTERRUTTORE REVERSE
16020 REM*******************************
16030 RV=1-RV : RETURN

17000 REM*******************************
17010 REM INTERRUTTORE FLASH
17020 REM*******************************
17030 FL=1-FL : RETURN
```

Commento

Linea 16030: Ogni volta che questa linea viene chiamata battendo "lira sterlina R", cambia il valore della variabile RV da zero a uno e viceversa. Nel modulo di stampa principale (3.1.5), lo stato di RV viene usato per determinare se un certo carattere deve essere riprodotto in negativo, cioè con il colore di sfondo e quello del carattere invertiti.

Linea 17030: Lo stesso processo per il lampeggiamento, attivato mediante la sequenza "lira sterlina F".

Verifica

Impartite il comando RUN e iniziate a creare un piccolo disegno, questa volta cercando però di attivare la visualizzazione in negativo e il lampeggiamento per alcuni caratteri con i comandi lira sterlina R o F.

Modulo 3.1.9: Controllo dei colori

Una volta imparato il modo per ottenere il lampeggiamento e la visualizzazione in negativo, possiamo, con questo modulo, stabilire il colore e la luminosità dei caratteri riprodotti sullo schermo, il colore dello sfondo e infine il colore del cursore e dei messaggi generati dal programma. Di solito non si ha bisogno di modificare il colore dei messaggi, ma è bene poterlo fare per certi casi limite, come ad esempio per distinguere i messaggi o il cursore su uno sfondo per il quale sia stato scelto il colore nero.

Modulo 3.1.9: Linee 14000-14230

```
14000 REM*******************************
14010 REM OPZIONI STAMPA
14020 REM*******************************
14030 IF A1$="C" THEN PROMPT$="COLORE TESTO
RIGA SUPERIORE"
14040 IF A1$="V" THEN PROMPT$="COLORE TESTO
RIGA INFERIORE"
14050 IF A1$="B" THEN PROMPT$="COLORE SFONDO
RIGA SUPERIORE"
14060 IF A1$="N" THEN PROMPT$="COLORE SFONDO
RIGA INFERIORE"
14070 IF A1$="P" THEN PROMPT$="COLORE CURSORE"
14080 FOR NP=1 TO 2
14090 COLOR 1,PC,PL
14100 PROMPT$="[RVS OFF][FLASH OFF]"+PROMPT$
14110 CHAR ,0,24,PROMPT$
14120 A1$="" : DO WHILE A1$<"1" OR A1$>"8"
14130 GOSUB 12000
14140 LOOP
14150 C(NP)=VAL(A1$)
14160 PROMPT$="LUMINOSITA'"
14170 GOSUB 24050
14180 NEXT NP
14190 C(2)=C(2)-1
14200 IF A2$="C" OR A2$="V" THEN C1=C(1) :
L1=C(2) : IF A2$="V" THEN C1=C1+8
14210 IF A2$="B" OR A2$="N" THEN C0=C(1) :
L0=C(2) : IF A2$="N" THEN C0=C0+8
14220 IF A2$="P" THEN PC=C(1) : PL=C(2)
14230 RETURN
```

Commento

Linee 14030-14070: Generano i messaggi corrispondenti al comando battuto dall'utente.

Linee 14080-14180: Questo ciclo provoca la generazione di due messaggi e l'acquisizione dei dati introdotti. Il primo, che viene preso dall'elenco fornito nelle linee 14030-14070, specifica le caratteristiche attivate, il secondo, generato al secondo passaggio attraverso il ciclo, serve a specificarne la luminosità.
Va notato che, nonostante i colori siano numerati da uno a quindici, il valore acquisito, sia per i colori che per la luminosità, è compreso fra 1 e 8. Per i colori del secondo gruppo, dall'arancione al verde chiaro, basta semplicemente specificarlo nel comando originale e poi premere il tasto numerico successivo a quello del quale compare il nome del colore. Nel caso del colore dei messaggi, sono disponibili solo i primi otto colori. Non appena vengono introdotti, i valori corrispondenti al colore vengono memorizzati nell'elemento uno del vettore C, quello corrispondente alla luminosità, nell'elemento C(2) (l'elemento zero non viene utilizzato).

Linee 14190-14220: Il valore della luminosità è ridotto di uno per poter tener conto del fatto che quest'ultima va da zero a sette invece che da uno a otto, e quindi i valori introdotti sono destinati a C1/L1, CO/LO e PC/PL, a seconda delle caratteristiche selezionate.

Verifica

Eseguite il programma come prima. Ora dovreste poter modificare i colori in cui compaiono i caratteri sullo schermo, il colore di fondo dello schermo e il colore nei punti in cui compaiono i messaggi e il cursore. Fate qualche esperimento con il programma per avere qualche idea in più sulle sue capacità.

Modulo 3.1.10: Memorizzazione dell'immagine su schermo

Ora che avete imparato a scrivere programmi che permettono di creare disegni sullo schermo, passiamo al problema della memorizzazione su disco del risultato ottenuto. Le soluzioni in questo caso sono due. La prima consiste nel tradurre il disegno in una stringa piuttosto complessa che può essere salvata sotto forma di serie di numeri a loro volta ritraducibili in un disegno. Il vantaggio di questo metodo è che una volta che essi sono stati richiamati da disco, è possibile conservare più disegni in memoria contemporaneamente e riprodurli sullo schermo, volendo, anche uno di fianco all'altro. L'aspetto indesiderato, invece, è rappresentato dal fatto che i disegni richiedono un certo tempo per essere caricati dal disco e il più grande elaborabile in questo modo non può superare i 240 caratteri (es.: un disegno di 20*12).
Il secondo metodo consiste nel salvare il contenuto dello schermo e la memoria colore sul disco come se fossero un normale file di programma. Questo significa

che è possibile trattare un solo disegno alla volta, ma i disegni possono essere trasferiti sullo schermo molto più rapidamente (non nel caso del nastro, però). L'aspetto negativo, in questo caso, è rappresentato dal fatto che con questo metodo è possibile salvare solo l'intera immagine su schermo e non parte di essa.
Poiché questo è il metodo più semplice, cominceremo a vedere come si salva un'immagine su schermo trattandola come un file di programma.
Quando il C16 riceve un'istruzione per salvare un programma, prima di tutto controlla le quattro locazioni di memoria (da 43 a 46), riservategli e note come "registri di sistema". Queste locazioni di memoria particolari registrano dove inizia e finisce in memoria il programma BASIC. Dopo aver stabilito questi due punti, il C16 salva semplicemente su disco il contenuto di tutti i byte di memoria che si trovano in mezzo, sotto forma di numeri. Per poter memorizzare l'immagine, quindi, tutto ciò che occorre fare è far credere al C16 che il programma inizia e finisce con l'immagine. In questo modo, qualsiasi istruzione di salvataggio del programma avrà come risultato la memorizzazione della sola immagine. Fatto questo, naturalmente, l'imbroglio deve poi essere eliminato rapidamente, altrimenti il C16 potrebbe confondersi e cercare di comportarsi come se l'immagine sullo schermo sia effettivamente il programma BASIC.

Modulo 3.1.10: Linee 18000-18150

```
18000 REM*********************************
18010 REM SAVE DEL VIDEO
18020 REM*********************************
18030 COLOR 1,PC,PL : CHAR ,0,23,"NOME DEL
DISEGNO" : INPUT Q$
18040 GOSUB 24040
18050 Q$="R0:"+Q$+"=VIDEO"
18055 POKE 3072+997,C0 : POKE 3072+998,L0
18060 FOR I=0 TO 3 : A%(I)=PEEK(43+I) : NEXT
18070 POKE 43,0 : POKE 44,8 : POKE 45,0 : POKE
46,16
18080 SAVE "@0:VIDEO",8
18090 POKE 43,A%(0) : POKE 44,A%(1) : POKE
45,A%(2) : POKE 46,A%(3)
18100 OPEN 15,8,15
18110 PRINT#15,Q$
18120 INPUT#15,A,B$ : CLOSE 15
18130 CHAR ,0,24,B$
18140 GOSUB 24030
18150 RETURN
```

Commento

Linee 18030-18040: Il programma chiede all'utente di indicare il nome con cui va salvata l'immagine su schermo.

Linea 18050: Questa strana aggiunta crea una stringa di questo tipo:

"RO:<NOMEFILE>=VIDEO"

che verrà usata in seguito come istruzione diretta al drive.

Linea 18055: Questa linea non è utilizzata da questo programma, ma viene inserita ad uso del prossimo, il programma DISPLAY. Essa ha a che fare con il fatto che l'immagine su schermo non contiene alcuna indicazione del colore di fondo originale e comparirà su quello che è selezionato nel momento in cui l'immagine verrà caricata. Queste due istruzioni POKE memorizzano i valori relativi al colore di fondo e alla luminosità nei due byte della memoria immagine che vengono normalmente usati per i messaggi (che sono vuoti). Non appena l'immagine su schermo viene ricaricata, essi possono essere letti (PEEK) per vedere quale dovrebbe essere il colore di fondo richiesto.

Dopo aver introdotto il programma DISPLAY, potrete aggiungere questa possibilità al programma ARTIST, anche se qui è meno necessaria dato che il colore di fondo può essere modificato abbstanza facilmente.

Linea 18060: Il contenuto attuale dei registri da 43 a 46 della memoria, che come sapete contengono la posizione iniziale e quella finale del programma BASIC, è conservato nel vettore A%.

Linea 18070: I valori conservati nei registri da 43 a 46 sono sostituiti con quelli nuovi che specificano l'inizio della memoria colore e la fine della memoria immagine (da 2048 a 4096). Il modo per calcolare i valori posti nei registri consiste nel trattarli come due coppie. In termini decimali, ciascuno dei due valori è costituito dal primo valore più 256 volte il secondo (es.: 0 + 256*8 = 2048).

Linea 18080: Creato il tranello per il C16, tutto ciò che occorre fare e salvare (SAVE) quello che crede sia il programma. Il nome attribuito non è quello specificato dall'utente, ma VIDEO. Fra poco vi spiegheremo perché.

Linea 18090: Nei registri dal numero 43 al numero 46 vengono riposti di nuovo i valori originali. Come potete notare, non possiamo usare un ciclo FOR...NEXT per raggiungere lo scopo, perché, anche se il C16 è in grado di ricordare dove era conservato il contenuto dei vettori, non avrebbe alcun senso per lui la variabile del ciclo (lo abbiamo momentaneamente confuso circa la posizione di qualsiasi valore in memoria).

Una volta che abbiamo riutilizzato l'istruzione POKE, comunque, tutto tornerà come prima.

Linee 18100-18130: Queste linee si riferiscono all'itnerazione fra il C16 e il drive e possono tranquillamente essere ignorate da chi usa le cassette.

Linee 18100-18120: La funzione di queste linee va accettata sulla fiducia, dato che non abbiamo intenzione di trattare l'argomento della gestione dei dischi (per saperne di più, vi consigliamo di leggervi il libro "Commodore 264/C16 Disk Companion di Mark England e David Lawrence, Sunshine Books, 1984). In breve, il metodo serve ad aprire un canale di comunicazioni speciale con il drive, noto come "canale di errore". Il C16 invia lungo questo canale un comando che attribuisce un nuovo nome al file memorizzato come file VIDEO e poi estrae il messaggio di errore del drive (linea 18120), che è semplicemente un OK, se tutto è andato come previsto. Se però il file non è stato salvato correttamente, per esempio perché ce n'è già uno con lo stesso nome o c'è un errore sul disco, l'utente riceve un messaggio di avvertimento.
Notate che l'operazione che usa il canale di errore verso il drive, viene deliberatamente usata al posto del comando RENAME e della variabile di sistema incorporata nel BASIC del C16, cioè DS$. In teoria, le linee 18100-18130 dovrebbero poter essere sostituite da:

```
18110 RENAME "VIDEO" TO (Q$)
18120 CHAR,0,24,DS$
```

In realtà, nell'usare RENAME o la variabile DS$, che contiene il messaggio di errore del drive, abbiamo riscontrato il verificarsi di un'interruzione nelle comunicazioni fra il drive e il C16. Questo libro è stato scritto, naturalmente, sulla versione più avanzata del C16. Per quando comprerete questo libro, la Commodore (così ci ha assicurato) avrà risolto i problemi relativi ai comandi e potrete sostituire le linee dalla numero 18100 alla numero 18130.
La ragione per cui il file è stato salvato come file VIDEO, è che mentre i puntatori di inizio e fine venivano modificati, il C16 non sarebbe stato in grado di ricordare che cosa è Q$. Il salvataggio come VIDEO, invece, ci consente di dare un nuovo nome al file una volta che la memoria è ritornata alla normalità. Se decidete di usare il comando RENAME, prendete nota dell'uso piuttosto insolito delle parentesi prima e dopo il nome della variabile di stringa che contiene il nome del file. Il comando RENAME non accetta nel comando una variabile di stringa come primo o secondo nome, a menoché non sia chiusa fra parentesi quadre.

Verifica

Eseguite il programma e create un disegno, possibilmente qualcosa di molto semplice, nel caso in cui ci sia un errore nel modulo di salvataggio dell'immagine su schermo e dobbiate reinserire il tutto di nuovo. Ora battete il tasto con la lira sterlina seguito dal tasto S. Battete un nome di file e premete il tasto RETURN. Il drive dovrebbe mettersi in funzione e dovreste veder comparire in basso sullo schermo il messaggio di errore "OK". Ora fermate il programma e, usando il comando DIRECTORY, controllate che sul disco ci sia un file di programma con il nome scelto e lungo nove blocchi.

Modulo 3.1.11: Ricaricamento da disco

Prima di esaminare il secondo metodo di salvataggio di un disegno, dobbiamo

introdurre il modulo di ricaricamento, che contiene sette linee, studiate appositamente per ricaricare un disegno salvato nel secondo formato, che sarebbero difficili da capire senza questi ultimi moduli. La ragione per cui questo modulo deve essere introdotto ora è che ci permette di ricaricare l'immagine salvata durante la verifica del modulo 3.1.10 e di controllare che quest'ultimo funzioni correttamente.

Modulo 3.1.11: Linee 23000-23170

```
23000 REM*******************************
23010 REM CARICA DAL DISKETTE
23020 REM*******************************
23030 COLOR 1,PC : CHAR ,0,23,"DISEGNO O VIDEO
(D/V)" : INPUT Q1$
23040 GOSUB 24040
23050 COLOR 1,PC : CHAR ,0,23,"NOME DEL FILE
DA CARICARE" : INPUT Q2$
23060 GOSUB 24040
23070 IF Q1$="D" THEN 23090
23080 LOAD Q2$,8,1
23090 Q2$=Q2$+",S,R"
23100 S1$="" : S2$=""
23110 OPEN1,8,2,Q2$
23120 INPUT#1,WW,SIZE,TX,TY,C0,L0
23130 FOR I=1 TO SIZE
23140 INPUT#1,TT : S1$=S1$+CHR$(TT)
23150 INPUT#1,TT : S2$=S2$+CHR$(TT)
23160 NEXT : CLOSE1
23170 GOSUB 21080 : RETURN
```

Commento

Linee 23030-23060: Il modulo chiede all'utente di specificare il formato da ricaricare, o come immagine completa o come vettore contenente parte di essa, e il nome sotto il quale deve avvenire il caricamento.

Linea 23070-23080: Queste due linee sono tutto ciò che occorre per ricaricare un'immagine su schermo, salvata come file di programma. Esse contengono però alcune cose interessanti da evidenziare. Innanzitutto non abbiamo resistito alla tentazione di usare un'istruzione GOTO, semplicemente perché è il metodo più facile per saltare da una parte all'altra del programma. Avremmo potuto predisporre un ciclo fittizio, come nel programma del grafico tridimensionale, ma la cosa sarebbe risultata francamente un po' troppo artificiosa.
In secondo luogo, osservate la forma del comando LOAD, e in particolare

l'indirizzo 1 alla fine. Si tratta di un'informazione fornita al C16 con cui gli si dice che, invece di caricare il file di programma su disco nel punto in cui andrebbe solitamente memorizzato un programma BASIC, deve caricarlo esattamente nella stessa area di memoria dalla quale è stato salvato con l'istruzione SAVE.
Tutti i file di programma su disco iniziano con due byte che registrano l'indirizzo di inizio del programma che è stato salvato con SAVE. Nel nostro caso, il programma iniziava da 248, corrispondente all'inizio della memoria colore, ed è qui che va ricaricato.
Un altro punto interessante da notare su queste linee è che non hanno niente che le chiuda: nessuna istruzione RETURN, ad esempio. Questo perché, quando il C16 riceve l'ordine di caricare (LOAD) un certo programma "mediante un programma", attua quella che viene chiamata "esecuzione automatica". In altre parole, non appena il nuovo programma è stato caricato, il programma BASIC corrente viene eseguito anche senza annullamento delle variabili in memoria. Nel nostro caso, dopo aver ricaricato l'immagine su schermo, siamo ancora alle prese con il nostro programa ARTIST, come programma in BASIC corrente, che riparte ancora dall'inizio. Adesso possiamo vedere lo scopo della variabile IN, definita nel primissimo modulo. Quando il programma viene eseguito dall'inizio, invece di aversi l'immediata cancellazione (da parte del modulo di inizializzazione dell'immagine su schermo che è già stata ricaricata, accade che il ciclo del modulo 3.1.1 riconosca che IN è stata aperta e salti in avanti fino al modulo 3.1.2).

Linee 23100-23170: Una routine di caricamento dei file di dati un po' più normale. L'elenco delle variabili ricaricate, e le due stringhe formate, si chiariranno meglio quando analizzeremo il secondo metodo di memorizzazione di una figura su schermo. Se volete ricaricare dei disegni in un altro programma usando il secondo formato di memorizzazione, è indispensabile che inseriate nell'altro programma queste linee. Fra parentesi, l'istruzione GOSUB nell'ultima linea è una chiamata ad un modulo che segue e che traccia sullo schermo la figura tratta dal disco.

Verifica

Supponendo che abbiate salvato un disegno durante la verifica dell'ultimo modulo, ora potete ricaricarlo eseguendo il programma e battendo il tasto con la lira sterlina seguito dal tasto L. Specificate che state ricaricando un'immagine su schermo e il nome del file e dovreste poi vedere ricomparire sullo schermo il disegno precedentemente salvato. Notate che, se per errore battete il nome di un programma reale su disco che è più lungo di 2048 caratteri, andrà sopra l'inizio del programma ARTIST e il C16 "scoppierà". Si fa per dire, in ogni caso vi toccherà premere il reset e ricaricare il programma.

Modulo 3.1.12: Definizione degli angoli di un disegno da memorizzare

Passiamo ora al secondo metodo di memorizzazione, consistente nel conservare un certo disegno sotto forma di serie di numeri memorizzati su disco, che possono essere tradotti in una stringa in grado a sua volta di ricreare sullo schermo un rettangolo con un massimo di 240 caratteri.

Modulo 3.1.12: Linee 19000-19070

```
19000 REM*******************************
19010 REM DEFINISCE GLI ANGOLI
19020 REM*******************************
19030 GETKEY T$
19040 IF T$<>"1" AND T$<>"2" THEN RETURN
19050 IF T$="1" THEN TY=Y1 : TX=X1 : YY=0
:XX=1
19060 IF T$="2" THEN BY=Y1 : BX=X1 : YY=1
19070 IF XX=0 OR YY=0 THEN RETURN
```

Commento

Linee 19030-19040: Dopo aver battuto D, per il comando DEFINE, grazie a queste due linee, l'utente ha la possibilità di battere 1 o 2 (notate che durante l'attesa scomparirà il cursore lampeggiante), 1 per indicare l'angolo in alto a sinistra del disegno da memorizzare, 2 per l'angolo in basso a destra (questo è tutto ciò che occorre fare per definire un rettangolo da memorizzare.

Linee 19050-19070: TY (Top Y), TX, BX (Bottom Y) e BX sono le coordinate dei due angoli. XX e YY servono a sapere se sono stati definiti entrambi gli angoli o meno. La definizione di un nuovo angolo in alto elimina quella dell'angolo in basso, quindi gli angoli devono essere sempre definiti nello stesso ordine: prima l'angolo 1 e poi l'angolo 2. Se i due angoli non sono ancora stati definiti, l'esecuzione ritorna all'unità principale del programma; se invece lo sono stati, essa continua con il modulo che segue.

Verifica

Battete questa linea provvisoria:

```
20000 STOP
```

ed eseguite il programma. Battete quindi il tasto con la lira sterlina e subito dopo il tasto D. Vedrete il cursore lampeggiante scomparire dallo schermo. Premete allora 1 e il cursore ricomparirà. Battete ancora una volta il tasto con la lira sterlina, seguito però questa volta dal tasto D e poi premete 2. Il programma dovrebbe interrompersi alla linea 20000. A questo punto potete, volendo, eseguire di nuovo il programma e verificare che la definizione dell'angolo due prima dell'angolo uno non provoca l'interruzione del programma che ricorda infatti solo l'angolo uno.

Modulo 3.1.13: Controllo del disegno da memorizzare

I disegni vengono memorizzati solo se le coordinate degli angoli sono state

definite correttamente e il rettangolo risultante non supera i 240 caratteri.
Il modulo che segue serve ad eseguire i controlli necessari.

Modulo 3.1.13: Linee 20000-20110

```
20000 REM*******************************
20010 REM CONTROLLO DIMENSIONI
20020 REM*******************************
20030 DO WHILE BX<TX OR BY<TY
20040 COLOR 1,PC,PL : CHAR ,0,24,"DEFINITO
ERRONEAMENTE"
20050 GOSUB 24030 : RETURN
20055 EXIT : LOOP
20060 WW=BX-TX+1 : HH=BY-TY+1
20070 SIZE=WW*HH
20080 DO WHILE SIZE>240
20090 COLOR 1,PC,PL : CHAR ,0,24,"TROPPO
GRANDE"
20100 GOSUB 24030 : RETURN
20105 EXIT : LOOP
20110 GOSUB 21000 : GOSUB 22000 : RETURN
```

Commento

Linee 20030-20050: Per essere considerato valido, l'angolo in basso deve essere sotto quello in alto e a destra rispetto ad esso. Se non lo è, il programma genera un messaggio di errore.

Linee 20060-20100: Queste linee ricavano la larghezza e l'altezza del rettangolo definito e controllano che la sua superficie non superi i 240 caratteri, provocando in caso contrario la generazione di un messaggio di errore.

Verifica

Battete questa linea provvisoria:

```
21000 STOP
```

Ora eseguite il programma e definite un rettangolo con l'angolo uno sotto o a destra dell'angolo due. Questo dovrebbe far sì che il programma vi informi dell'errore di definizione. Riprovate, questa volta, però, ponendo l'angolo superiore del rettangolo nell'angolo in alto a sinistra dello schermo. Ora spostate il cursore in giù di 15 posizioni e a destra di altrettante e definite l'angolo due. Dovreste essere informati del fatto che il rettangolo è troppo grande.
Per finire, spostate il cursore in alto a sinistra di uno o due spazi e ridefinite il

secondo angolo (il programma dovrebbe fermarsi informando l'utente che tutte le prove sono state eseguite con successo).

Modulo 3.1.14: Trasformazione del disegno in una stringa

Lo scopo di questo modulo è di prendere il contenuto della memoria immagine e della memoria colore corrispondente al disegno che deve essere definito e di tradurre i valori delle due aree in due stringhe, con il valore ASCII di ciascun carattere a rappresentazione del contenuto di un byte. Il motivo della traduzione dei valori in una stringa, anche se, come abbiamo già detto, saranno memorizzati su disco come numeri, è che esiste un programma, che vedremo più avanti, in grado di memorizzare in modo più semplice e rapido questa stringa che non l'insieme elevato dei numeri memorizzati. I numeri vengono usati solo a vantaggio dei dispositivi di memorizzazione, in quanto certi caratteri di stringa non riproducibili in stampa non possono essere ricaricati da disco o nastro (vanno persi nel corso dell'elaborazione).

Modulo 3.1.14: Linee 21000-21140

```
21000 REM*********************************
21010 REM CREA LA STRINGA MAPPA VIDEO
21020 REM*********************************
21030 S1$="" : S2$=""
21040 FOR I=TY TO BY : FOR J=TX TO BX
21050 S1$=S1$+CHR$(FNCV(X))
21060 S2$=S2$+CHR$(FNCC(X))
21070 NEXT J,I : PRINT "s";
21080 SP=3072+40*TY+TX : SC=SP-1024
21090 COLOR 0,C0,L0
21100 FOR I=0 TO SIZE/WW-1
21110 FOR J=1 TO WW
21120 POKE SP+40*I+J-1,ASC(MID$(S1$,I*WW+J,1))
21130 POKE SC+40*I+J-1,ASC(MID$(S2$,I*WW+J,1))
21140 NEXT J,I
21150 RETURN
```

Commento

Linea 21030: Le due stringhe che verranno usate: S1$ per la memoria dell'immagine e S2$ per la memoria colore.

Linee 21040-21070: Usando le coordinate degli angoli superiore ed inferiore e le due funzioni definite dall'utente nel modulo 3.1.1, questo ciclo ricava il valore di tutti i caratteri del rettangolo specificati e aggiunge sotto forma di caratteri i valori e il relativo colore alle due stringhe.

Linea 21080: La posizione dell'angolo superiore del disegno nella memoria immagine e in quella del colore.

Linee 21100-21140: Queste linee non sono strettamente necessarie per questo programma, ma possono esserlo invece per sfruttare al massimo il programma ARTIST come strumento di lavoro. La loro funzione consiste nel prendere le du stringhe e nel tradurle nel disegno originale. In questo punto del programma, esse servono semplicemente a garantire che nella traduzione vada tutto bene, ma possono essere tolte dal loro contesto e usate in qualsiasi programma che ricarica un disegno da disco. Esse servono a passare in rassegna la stringa, rappresentando righe e colonne con due cicli, e a scrivere (POKE) il valore di ciascun carattere in un apposito punto della memoria dell'immagine o del colore. Risultato del loro impiego è la comparsa del disegno contenuto nel rettangolo specificato.

Verifica

Battete la linea provvisoria:

```
22000 STOP
```

eseguite il programma e create un piccolo disegno, definendo il rettangolo che lo contiene. Dopo pochi secondi dovreste vedere lo schermo azzerarsi e il disegno comparire di nuovo. Non appena è completo, il programma si fermerà.

Modulo 3.1.15: Memorizzazione del disegno

Dopo aver creato le stringhe necessarie a trasferire il disegno in memoria e dopo averle sottoposte a verifica tracciando di nuovo il disegno, possiamo passare alla memorizzazione dei dati sul disco. Il modulo non contiene niente di particolarmente complesso: esso memorizza semplicemente le variabili relative alla posizione e alle dimensioni dell'immagine su schermo e poi il contenuto delle due stringhe.

Modulo 3.1.15: Linee 22000-22110

```
22000 REM*******************************
22010 REM SALVATAGGIO DELLA STRINGA
22020 REM*******************************
22030 COLOR 1,PC,PL : CHAR ,0,23,"NOME DEL
DISEGNO" : INPUT Q$
22040 GOSUB 24040
22050 Q$="@0:"+Q$+",S,W"
22060 OPEN 1,8,2,Q$
22070
PRINT#1,WW;R$;SIZE;R$;TX;R$;TY;R$;C0;R$;L0
22080 FOR I=1 TO SIZE
22090 PRINT#1,ASC(MID$(S1$,I,1))
```

```
22100 PRINT#1,ASC(MID$(S2$,I,1))
22110 NEXT : PRINT#1 : CLOSE1 : RETURN
```

Verifica

Eseguite il programma e create un disegno. Definite un rettangolo che lo contenga e, dopo averlo elaborato e ridisegnato sullo schermo, il programma vi chiederà di specificare un nome di file. Appena lo farete, dovreste veder partire il drive cui è destinato il file contenente i dati. Appena il processo di memorizzazione è terminato, azzerate lo schermo battendo il comando "Lira sterlina [CLEAR], battete il comando di caricamento, "Lira sterlina L" e specificate il nome di file che avete assegnato al disegno. A questo punto dovreste veder ricomparire il disegno.

Conclusioni

Dopo aver introdotto quello che a molti di voi sarà certamente sembrato il programma più complesso mai usato, va sottolineato che esso ha senso e può valere tanta fatica solo se viene usato molto. Anche se può offrire motivo di divertimento di per sé, può essere di grande soddisfazione quando comincia a dare qualcosa di più a tutti i programmi in cui viene inserito, fornendo un metodo rapido ed efficiente per creare disegni su schermo abbastanza spettacolari. Il programma che segue è stato studiato per dimostrarvi quanto può essere facile usarlo per inserire delle figure in un programma: il resto, alla vostra immaginazione.

PROGRAMMA 3.2: DISPLAY

Dopo aver evidenziato l'utilità del programma ARTIST come strumento di lavoro, ci è sembrata una buona idea darvi qualche spunto su come usarlo. Esso offre un metodo di lavoro tanto semplice quanto efficace. Il suo scopo è visualizzare una serie di immagini su schermo, create dal programma ARTIST in successione continua.

A vedere certi risultati che possono essere ottenuti con questo programma, viene davvero voglia di usarli in pubblico, magari nella vetrina di un negozio o in un'esposizione. Chissà che non abbiate l'occasione di farlo.

Modulo 3.2.1: Nomi di file

I nomi dei file da caricare come immagini su schermo devono essere registrati in questo modulo, ciascuno come linea del vettore FI$. Notate che tutti iniziano da FI$(1), non da FI$(0) e finiscono con END. Non dimenticate che, se dovete usare più di otto nomi di file, dovete assegnare le necessarie dimensioni a FI$.

Modulo 3.2.1: Linee 1000-1080

```
1000 REM*********************************
1010 REM NOMI VIDEO
1020 REM*********************************
```

```
1030 FI$(1)="VIDEO 1"
1040 FI$(2)="VIDEO 2"
1050 FI$(3)="VIDEO 3"
1060 FI$(4)="VIDEO 4"
1070 FI$(5)="VIDEO 5"
1080 FI$(6)="FINE"
```

Modulo 3.2.2.: Caricamento delle immagini su schermo

Questo modulo scorre lungo tutti i nomi del vettore FI$ finché non trova la parola END e poi ricomincia da capo.

Modulo 3.2.2: Linee 2000-2130

```
2000 REM*********************************
2010 REM MODULO DI CONTROLLO
2020 REM*********************************
2030 TRAP 2100
2040 IF IN<>0 THEN COLOR
0,PEEK(3072+997),PEEK(3072+998)
2050 POKE 3072+997,32 : POKE 3072+998,32
2060 IF IN<>0 THEN FOR I=1 TO 20000 : NEXT I
2070 IN=IN+1
2080 IF FI$(IN)="FINE" THEN RUN
2090 SCNCLR : LOAD FI$(IN),8,1
2100 SCNCLR
2110 COLOR 0,1 : COLOR 1,8
2120 PRINT ERR$(ER),EL
2130 END
```

Commento

Linee 2040-2050: Questo è lo scopo che sta dietro la scrittura (tramite POKE) del valore del colore di fondo e della luminosità in un'estremità della memoria immagine nel modulo di salvataggio dell'immagine, incluso nel programma ARTIST. Se IN non è uguale a zero, il file deve essere caricato e il colore di fondo e la luminosità possono essere ricavati dai byte. Essi vengono poi scritti con POKE con il codice per uno spazio.

Linea 2060: L'immagine su schermo viene visualizzata per il periodo di tempo stabilito da questo ciclo (nel sottoporre a verifica il programma, potete abbreviare il ciclo in modo che lavori più rapidamente).

Linea 2070: Questa volta il valore di IN non registra solo il fatto che il programma è già inizializzato, ma anche quante volte è stato eseguito automaticamente.

Linea 2080: L'istruzione RUN, a differenza dell'esecuzione automatica, avvia il programma proprio dall'inizio e quindi torna alla prima immagine su schermo.

Verifica

Prima di poter sottoporre il programma a verifica, è necessario memorizzare, usando il programma ARTIST, alcuni disegni da usare poi. Probabilmente il modo più facile è quello di creare degli schermi vuoti con VIDEO 1,2,3,... nell'angolo in alto a sinistra, ciascuno con un colore di fondo diverso. Fatto questo, chiamate DISPLAY ed eseguitelo. Dovreste vedere i vostri schermi memorizzati nell'ordine in cui sono stati specificati nel modulo 3.2.1. Notate che se, alla linea 2060, usate un ciclo di temporizzazione molto corto, avrete poco tempo per distinguere i singoli colori nel passaggio dall'uno all'altro.

PROGRAMMA 3.3: CHARACTERS

Dopo aver parlato di un aspetto della grafica a bassa risoluzione, possiamo cominciare a parlare di qualcosa d'altro, un po' più insolito. Intendiamo parlare di un programma che permetta di modificare la forma dei caratteri che il C16 genera sullo schermo. Prima però di passare al programma vero e proprio, è necessario parlare del modo in cui i caratteri vengono creati e generati sullo schermo nel modo a bassa risoluzione.

Lo schermo a bassa risoluzione del C16 ha spazio per 25 linee di 40 caratteri ciascuna per un totale di 1000 spazi-carattere. In altri termini, è possibile generare fino a 1000 caratteri (molti dei quali uguali fra loro, visto che il C16 non è in grado di generare nello stesso momento 1000 caratteri diversi l'uno dall'altro). Questo 1000, però, non è tutto.

Se proviamo a dare un'occhiata da vicino ai caratteri sullo schermo, ci accorgiamo immediatamente che non sono formati da una linea continua, come quelli che state leggendo, ma da una serie di punti. In effetti, ciascuno dei 1000 spazi-caratteri che possono stare sullo schermo è costituito da 64 punti ed è la combinazione di questi 64 punti che costituisce i singoli caratteri che il C16 è in grado di generare sullo schermo. La lettera A, ad esempio, è ottenuta in questo modo:

Figura 3.1: Ingrandimento della lettera A come appare sullo schermo

```
                ====================
RIGA  1 --> !        0 0         !
      2 --> !      0 0 0 0       !
      3 --> !    0 0     0 0     !
      4 --> !    0 0 0 0 0 0     !
      5 --> !    0 0     0 0     !
      6 --> !    0 0     0 0     !
      7 --> !    0 0     0 0     !
      8 --> !                    !
                ====================
```

I punti che costituiscono i caratteri sono noti come "pixel" (un termine che sta per elementi dell'immagine, in inglese "picture element") e costituiscono la più piccola unità gestibile del C16, sia in alta che in bassa risoluzione.
Queste figure complesse non compaiono sullo schermo per caso: è chiaro che esiste qualcosa che va predisposto nella memoria del computer in modo che, premendo il tasto A, ad esempio, compaia sullo schermo la configurazione di punti che crea quella lettera. In realtà, tutti i caratteri che il C16 è in grado di riprodurre sullo schermo sono memorizzati sotto forma di numeri nel blocco di memoria che inizia dall'indirizzo 53248 e che è chiamato "memoria dei caratteri" o "ROM dei caratteri". Ad ogni carattere disponibile sono riservati otto byte di memoria, ciascuno dei quali stabilisce dove devono comparire, su ogni linea, i pixel che costituiscono il carattere. Questo viene ottenuto costruendo un'immagine del carattere con gli uno e gli zeri del sistema binario usato dal C16, dove i numeri sono espressi in termini di potenza del numero due invece che del numero dieci, come nei conteggi normali. Così:

2013006

nel nostro sistema di numerazione normale significa:

(2*10^6)+(0*10^5)+1*10^4)+(3*10^3)+(0*10^2)+(0*10^1)+(6*10^0),

mentre nel sistema binario, dove le uniche cifre consentite sono uno e zero, un numero come:

11001010

significa:

(2^7)+(2^6)+(2^3)+(2^1) (ignorando gli zeri).

Non ci interessa che conosciate perfettamente l'aritmetica binaria, purché ricordiate che un byte di memoria nel C16 è in grado di contenere un numero binario di otto cifre e che tutti gli zeri e gli uno sono un modo perfetto per registrate quali pixel devono essere usati e quali no in ogni riga. La lettera A, per esempio, è costituita da questi otto numeri binari:

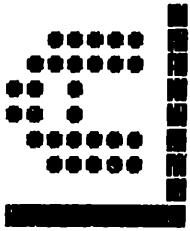
```
0 0 0 1 1 0 0 0
0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 0
0 1 1 1 1 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0
```

Se fate attenzione, potrete riconoscere la forma della A, questa volta data dagli 1 invece che dai punti. Più difficile, invece, vi sarà riconoscere in essi i numeri:

24, 60, 102, 126, 102, 102, 102, 0

cioè quello che i numeri binari diventano se tradotti nel più comodo sistema di numerazione decimale.
Tutto quanto abbiamo detto serve solo a sottolineare il fatto che, ad intervalli regolari, il C16 guarda il contenuto della memoria dell'immagine, che abbiamo già usato per scrivere (POKE) i caratteri sullo schermo, prende il numero di ogni carattere che dovrebbe essere sullo schermo, lo cerca nella memoria dei caratteri e usa ciò che trova per disegnare il carattere sullo schermo.
Questo processo continua ad essere eseguito anche quando sembra che lo schermo non subisca variazioni.
Da tutto ciò si deduce che se fosse possibile modificare il contenuto della memoria dei caratteri, potremmo modificare la forma dei caratteri sullo schermo, potremmo avere caratteri scelti da noi, nuovi caratteri grafici e qualsiasi cosa possa essere ricavata da un quadrato di 8*8 punti.
C'è un problema però, ed è che la memoria dei caratteri non può essere modificata. Fa parte della ROM, cioè di quella memoria a sola lettura, incorporata nel C16. Ciò che possiamo fare, quindi, è copiarlo da qualche parte nella RAM, cioè in quel tipo di memoria che può essere modificata, e dire al C16 che quello è il luogo in cui deve andare a cercare i dati relativi ai caratteri da riprodurre sullo schermo. Fatto questo, possiamo giocare con il nostro set di caratteri quanto più ci piace (questo è, del resto, lo scopo di questo programma.

Figura 3.1: Caratteri



A

(quello che vedete è lo schermo durante il modo di editing dei caratteri, con la lettera A ruotata di 90 gradi).

```
CODICE CARATTERE: 1
<I> INVERSIONE
<M> RIFLESSO
```

```
<R> RITORNA ALLA GRIGLIA
<1> COLORA IL QUADRO
<0> SVUOTA IL QUADRO
<T> RUOTA IL CARATTERE
<P> LO PONE NELLA MEMORIA
<S> SALVA IL SET IN UN FILE
<L> CARICA UN SET DAL DISCO
<X> RITORNA AL SET DELLA ROM E FINE
```

Modulo 3.3.1: Un importante avvertimento

Non c'è niente di particolarmente difficile in questo modulo: esso esiste infatti unicamente per ricordare che, prima di usare questo programma, bisogna trasferire i caratteri dalla ROM, dove non possono essere modificati, alla RAM, dove è consentita qualsiasi modifica.
Quest'ultima non è apportata dal programma stesso, in parte perché richiederebbe un mucchio di tempo e in parte per il fatto che il comando PEEK del BASIC sul C16 non riconosce l'esistenza degli indirizzi di memoria fuori dal campo compreso fra 1 e 16384, e, per copiare la memoria, dovremmo far uso del comando PEEK. Molto più semplice, in questo caso, usare il monitor del codice macchina incorporato nel C16, che dispone di un comando che esegue il trasferimento del set di caratteri quasi instantaneamente.
Introduciamo il monitor battendo semplicemente:

MONITOR[RETURN]

facendolo seguire da:

T D000 D7FF 3800[RETURN]

I due comandi hanno l'effetto di copiare 2048 byte di memoria a cominciare dall'indirizzo 53248 della ROM, in un nuovo indirizzo a partire da 14336. Questo permette di porre il set di caratteri in cima alla memoria disponibile sul C16, lontano dal programma BASIC. Resta comunque il pericolo che, nel caso vogliate ampliare il programma in modo consistente, i nuovi ampliamenti vadano a scriversi sopra il set di caratteri.
Va detto che, con quanto abbiamo descritto, abbiamo trasferito solo un set, quello dei caratteri maiuscoli. Per trasferire tutti e due i set, occorrerebbero 4096 byte di memoria (4K), cosa questa che porterebbe all'uso di parte della memoria a nostra disposizione. Nell'utilizzare il nostro set di caratteri personalizzato, quindi, avremo a nostra disposizione solo i caratteri maiuscoli (il voler passare a quelli minuscoli creerebbe un pasticcio).

Dopo aver trasferito il blocco di memoria, battete:

X[RETURN]

per uscire dal monitor e ritornare al BASIC. Il procedimento può essere attuato sia prima che dopo il caricamento del programma BASIC, quindi non preoccupatevi se lo eseguite e poi vi rendete conto di non aver trasferito il set di caratteri. Semplicemente rispondete N alla richiesta del programma ed introducete il monitor. Il programma BASIC non subirà alcuna conseguenza.

Modulo 3.3.1: Linee 10000-10210

```
10000 REM*******************************
10010 REM AVVERTENZA
10020 REM*******************************
10025 DO UNTIL IN : IN=1
10030 SCNCLR : COLOR 0,2,7 : COLOR 1,3,4
10040 PRINT
10050 PRINT"              AVVERTIMENTO"
10060 PRINT
10070 PRINT "QUESTO PROGRAMMA MANDERA' IN
BLOCCO IL"
10080 PRINT "   C16 SE IL SET DEI CARATTERI
NON"
10090 PRINT"       VIENE TRASFERITO NELLA RAM"
10100 PRINT
10110 PRINT" PER FAR CIO' ESEGUI I SEGUENTI
COMANDI"
10120 PRINT"           DIRETTAMENTE IN BASIC"
10130 PRINT
10140 PRINT"                  MONITOR"
10150 PRINT"             T D000 D7FF 3800"
10160 PRINT"                       X
10170 PRINT
10180 PRINT"SOLO ALLORA POTRAI DARE IL RUN A
QUESTO"
10190 PRINT"                 PROGRAMMA"
10195 INPUT "[CD][CD][CD]HAI TRASFERITO I
CARATTERI (S/N):"; Q$
10200 IF Q$<>"S" THEN STOP
10210 LOOP
```

Modulo 3.3.2: Selezione del nuovo set di caratteri

Dopo aver trasferito la memoria caratteri sulla RAM, è ora necessario dire al C16 che deve prendere i suoi dati dal nuovo set appena trasferito e inizializzare un certo numero di variabili.
È importante che la tecnica utilizzata in questo modulo vi sia chiara. Una volta che avete ridisegnato il vostro set di caratteri usando questo programma, sarete liberi di usarlo per tutti i programmi, solo se inserirete questa routine per informare il C16 del fatto che non state usando il set di caratteri standard.

Modulo 3.3.2: Linee 11000-11150

```
11000 REM*******************************
11010 REM INIZIALIZZAZIONE
11020 REM*******************************
11030 TRAP 11110
11040 IF IN=0 THEN CH=0 : DIM TT%(7,7) : IN=1
11050 MENU$="IMTPSL"
11060 TT=DEC("FF12")
11070 POKE TT,PEEK(TT)AND251
11080 POKE TT+1,14*4
11090 POKE52,55 : POKE 56,55
11100 GOSUB 12000
11110 POKE TT+1,208
11120 POKE TT,PEEK(TT) OR 4
11130 SCNCLR
11140 IF BB<>1 THEN PRINT ERR$(ER),EL
11150 END
```

Commento

Linea 11030: Notate che il comando TRAP non si limita a puntare alle linee che azzerano lo schermo e a generare il messaggio di errore corrente, ma salta a metà del modulo, alle linee che, come vedremo, ripristinano il set di caratteri prima della fine del programma.

Linea 11040: Il vettore TT% viene usato per memorizzare la posizione dei pixel di un singolo carattere.

Linee 11060-11080: Queste sono le linee che selezionano il nuovo set di caratteri. In pratica esse modificano il contenuto dei due registri della memoria del C16, che vengono usati per stabilire se il set di caratteri deve essere cercato nella ROM o nella RAM, e in che punto della memoria si trova. L'indirizzo del primo dei due registri è 65298 o FF12 nel sistema esadecimale. La linea 11070 dice al C16 di cercare il set di caratteri che si trova nella RAM, mentre la linea 11080,

lo informa che l'indirizzo iniziale del set di carattere è 14*4*256 o 14336. Ogni volta che si seleziona e si usa un set di caratteri personalizzato, queste sono le linee che consentono di farlo.

Linea 11090: Questa linea pone un limite nella memoria usata dal programma BASIC (il registro all'indirizzo 56) ed assicura inoltre che la memoria di stringa, che si trova di solito all'indirizzo più alto disponibile, non vada a scriversi sopra il set di caratteri. Per entrambi gli scopi, al C16 viene detto che il più alto indirizzo di memoria disponibile è 14335. È fondamentale che esso sia introdotto in memoria prima del caricamento del programma, dato che può verificarsi qualche errore negli ultimi due caratteri del set, derivante dalla memorizzazione di Q$ nel modulo precedente. Tocca a voi decidere se ci possono essere dei problemi.

Linee 11110-11120: Il set di caratteri standard viene ripristinato dicendo al C16 di cercarlo all'indirizzo 208*256 (53248) della ROM. Queste sono le linee che vanno sempre usate quando si pone fine ad un programma che ha usato un set di caratteri personalizzato. Esiste un certo numero di casi, per altro molto limitato, in cui queste linee possono non funzionare correttamente, a causa dei complessi meccanismi di temporizzazione del C16, e il set di caratteri personalizzato rimane attivo anche dopo la fine del programma. In questo caso, per riportare la situazione alla sua normalità, è necessario rieseguire il programma e premere STOP. Esistono ancora altri casi in cui quanto detto può provocare un blocco del C16 con la comparsa di un'immagine "inquietante". Nessuna delle due situazioni si verifica di frequente. Non c'è comunque da preoccuparsi: basta premere il tasto di reset e tutto torna normale.
Un'altra cosa che è indispensabile ricordare è la necessità di salvare il programma completamente prima di sottoporre a verifica i singoli moduli. Se dovete premere il tasto di reset prima di aver salvato l'ultimo modulo, il contenuto di quest'ultimo va perso.

Verifica

La verifica di questo modulo è davvero molto elementare. Per prima cosa, assicuratevi di aver eseguito il trasferimento evidenziato nel commento al modulo 3.3.1. Poi battete la linea provvisoria:

```
12000 PRINT "CIAO": GETKEY A$:RETURN
```

ed eseguite il programma. Rispondete Y al sollecito e vedrete comparire la parola inserita nella linea. Quando siete soddisfatti del risultato, premete un tasto qualsiasi e il programma avrà termine (non preoccupatevi dei messaggi di errore generati dalla linea 11140). Siete tornati così al set di caratteri standard. Se qualcosa non ha funzionato, ve ne accorgerete subito perché lo schermo sarà un vero guazzabuglio. Per azzerarlo, premete il tasto reset e ripetete tutto da capo.

Modulo 3.3.3: Visualizzazione di un carattere ingrandito

La funzione di questo programma è permettere di modificare i caratteri molto facilmente. Un modo in cui ciò può essere ottenuto è generando la versione ingrandita di un certo carattere in modo da poterci spostare sopra il cursore. Questo modulo permette per l'appunto di specificare il carattere e poi di generarlo.

Modulo 3.3.3: Linee 12000-12190

```
12000 REM*******************************
12010 REM STAMPA LA GRIGLIA CARATTERE
12020 REM*******************************
12030 COLOR 0,1 : COLOR 4,3,2
12040 DO : COLOR 1,3,6
12050 SCNCLR : FOR I=0 TO 7 : CHAR ,8,I,"[RVS
ON] " : NEXT : PRINT
12060 PRINT "[RVS ON]          "
12070 CP=14*1024+CH*8
12080 COLOR 1,6,4 : PRINT"[HOME]"; : FOR I=CP
TO CP+7 : FOR J=7 TO 0 STEP-1
12090 IF (PEEK(I) AND 2↑J)=2↑J THEN PRINT
CHR$(113); : ELSE PRINT"[RVS OFF] ";
12100 NEXT J : PRINT
12110 NEXT I
12120 T=40*4+15 : POKE 3072+T,CH : POKE
2048+T,106
12130 COLOR 1,8,6 : PRINT "[CD][CD]CODICE
CARATTERE:";CH
12140 INPUT "[CD]NUMERO PER IL CAMBIO
(0=MODIFICA):";MM:CH=CH+MM
12150 IF CH<0 THEN CH=0
12160 IF CH>255 THEN CH=255
12170 IF MM=0 THEN GOSUB 13000
12180 LOOP
12190 RETURN
```

Commento

Linee 12050-12060: Le linee disegnano il perimetro di un quadrato di 8*8 nell'angolo in alto a sinistra dello schermo, usando spazi visualizzati in negativo.

Linea 12070: La variabile CP viene usata per registrare l'indirizzo iniziale in memoria del gruppo di otto byte che registrano i dettagli del carattere corrente, rappresentato da CH, che inizia da zero o dal simbolo "@".

Linee 12080:12110: Questi due cicli analizzano tutte le cifre binarie di tutti gli otto byte che registrano la forma del carattere. L'operatore AND viene usato per stabilire se una certa cifra è 1o 0. Se volete capire meglio l'uso di AND, provate ad eseguire questo esperimento. Battete:

```
BI=225[RETURN]
```

che attribuisce alla variabile BI il valore uguale al numero che, in notazione binaria, sarebbe 1110001, cioè 128+64+32+, guardando le cifre 1. Ora battete:

```
PRINT 1 AND BI
```

Otterrete come risposta un 1. Battete allora:

```
PRINT 32 AND BI[RETURN]
PRINT 64 AND BI[RETURN]
PRINT 128 AND BI[RETURN]
PRINT 16 AND BI[RETURN]
```

A queste linee, il programma risponderà fornendo, come risultato i numeri 32, 64, 128 e 0. Cosa è successo? Ogni volta che viene introdotto un numero da sommare con AND a BI, questi viene confrontato con 255 nella forma binaria, ad esempio:

```
32 AND 255
```

fa':

```
00100000 AND 11100001
```

e ciò che viene fornito è un numero binario in cui vengono sommate solo quelle cifre che sono 1 in entrambi i numeri. Negli esempi appena introdotti, i numeri sommati con AND a BI hanno una sola cifra binaria uguale a 1, dato che sono tutti potenze di due, nello stesso modo in cui 1.000.000 nel nostro sistema decimale ha una sola cifra diversa da zero, dato che è una potenza di 10. Se in BI, la cifra binaria corrispondente fosse 1, si avrebbe lo stesso valore. In quest'ultimo caso (16AND225), comunque, la cifra che rappresenta 16 viene posta a 0 in BI e quindi AND registra la mancanza di corrispondenza.
Nel programma, il valore della variabile J viene usato per creare gli otto elevamenti a potenza di due che possono essere contenuti in un unico byte e poi per vedere se sono presenti nel byte esaminato con AND. Se nel quadrato 8*8 viene riscontrato un 1, viene stampato un punto ingrandito, che rappresenta un pixel, se invece l'uno non c'è, viene prodotto uno spazio. In questo modo, si ottiene la generazione sullo schermo della versione ingrandita del carattere registrato.

Linea 12120: Questa linea scrive nella memoria immagine il codice del carattere in fase di elaborazione e seleziona il giallo per il byte colore corrispondente.

L'istruzion POKE è molto più semplice dell'istruzione CHAR, in quanto la variabile CH contiene un codice schermo, non un codice ASCII (per sapere qual è la differenza, rileggete le due appendici nel manuale dell'utente) e la stampa di CHR$(CH) non produce un risultato corretto. La versione del carattere nelle sue dimensioni normali compare alla destra del quadrato 8*8.

Linee 12140-12170: Il carattere sul video può essere modificato mediante somma o sottrazione dal valore di CH, entro un campo di codici di carattere compresi fra 0 e 255. Notate che i codici di carattere, cui si fa riferimento qui, non sono i codici ASCII. Non li troverete cioè nella tabella del vostro manuale, dove sono invece riportati diversi caratteri che non sono caratteri veri e propri, ma brevi istruzioni al C16 e non possono essere riprodotti su schermo. Quelli che vedete qui sono i codici di schermo, riportati anch'essi in una tabella e che sono tutti riproducibili. Per spostarvi da un numero all'altro, basta battere un numero da aggiungere o sottrarre al valore corrente di CH (ad esempio 10 o -10).
Notate che il C16 ricorda sempre l'ultimo valore introdotto in modo che se, ad esempio, battete 1, l' uno può essere riprodotto semplicemente premendo il tasto RETURN.

Verifica

Potete eseguire il programma finché avete trasferito il set di caratteri. Nell'angolo in alto a sinistra dovreste vedere la versione ingrandita del simbolo "@" e dovreste inoltre essere in grado di spostarvi lungo il set di caratteri, esaminando qualsiasi carattere a scelta. Ciò che non potete ancora fare è intervenire sul carattere visualizzato. Questo potrete farlo solo in seguito, premendo il tasto 0 e chiamando con esso un modulo che vedremo più avanti. Normalmente, la scelta del modo 0 provoca l'arresto del programma, ma in questo modulo la cosa è possibile purché spostiate il puntatore CH e premiate STOP prima che sia disegnato il nuovo carattere. Premendo STOP mentre il programma è in attesa di un input, non si ottiene alcun effetto.

Modulo 3.3.4: Il cursore lampeggiante

Questo è un modulo molto semplice e lineare, simile ad uno dei moduli che abbiamo esaminato nel corso del programma ARTIST. La sola vera differenza è che ci sono solo due possibilità da ricordare quando si sostituisce il carattere originale dopo ogni lampeggiamento, vale a dire cerchi verdi o spazi vuoti.

Modulo 3.3.4: Linee 14000-14100

```
14000 REM*******************************
14010 REM LAMPEGGIO CURSORE
14020 REM*******************************
14030 CC=PEEK(3072+40*Y1+X1)
14040 A$="" : DO UNTIL A$<>""
14050 CHAR ,X1,Y1,"[PINK]*"
```

```
14060 FOR I=1 TO 100 : NEXT
14070 COLOR 1,6,4 : CHAR ,X1,Y1,"" : IF CC=81
THEN PRINT CHR$(113) : ELSE PRINT " "
14080 FOR I=1 TO 50 : NEXT
14090 GET A$ : LOOP
14100 RETURN
```

Verifica

Battete:

RUN 14000[RETURN]

e vedrete comparire un asterisco lampeggiante nell'angolo in alto a sinistra dello schermo. Premete un tasto e il programma si fermerà con il messaggio di errore RETURN WITHOUT GOSUB.

Modulo 3.3.5: Introduzione dei comandi

Questo modulo è simile ad altri già introdotti nel programma ARTIST. La sua funzione consiste nel presentare un menu, spostare il cursore lampeggiante, riempire o cancellare i pixel ingranditi e accettare i comandi per la manipolazione del carattere corrente. Le istruzioni complete per il suo impiego sono contenute nel menu.

Modulo 3.3.5: Linee 13000-12230

```
13000 REM*******************************
13010 REM MANIOLAZIONE CARATTERI
13020 REM*******************************
13030 X1=0 : X2=0 : Y1=0 : Y2=0
13040 PRINT"[CU]
          [CU][CU][CU]":REM 40 SPAZI
13050 COLOR 1,16 : PRINT "<I>
INVERSIONE",,,"<M> RIFLESSO",,,"<R> RITORNA
ALLA GRIGLIA"
13060 PRINT"<1> COLORA IL QUADRO" : PRINT "<0>
SVUOTA IL QUADRO"
13070 PRINT"<T> RUOTA IL CARATTERE" :
PRINT"<P> LO PONE NELLA MEMORIA"
13080 PRINT"<S> SALVA IL SET IN UN
FILE":PRINT"<L> CARICA UN SET DAL DISCO"
13100 PRINT"<X> RITORNA AL SET DELLA ROM E
FINE"
```

```
13110 PRINT"[HOME]";
13120 A$="" : DO UNTIL A$="R"
13130 GOSUB 14000
13140 IF X1>0 AND A$="[CL]" THEN X1=X1-1 :
LOOP
13150 IF X1<7 AND A$="[CR]" THEN X1=X1+1 :
LOOP
13160 IF Y1>0 AND A$="[CU]" THEN Y1=Y1-1 :
LOOP
13170 IF Y1<7 AND A$="[CD]" THEN Y1=Y1+1 :
LOOP
13180 IF A$="1" THEN CHAR ,X1,Y1,CHR$(113)
13190 IF A$="0" THEN CHAR ,X1,Y1," "
13200 Z=INSTR(MENU$,A$)
13210 ON Z GOSUB
15000,16000,17000,18000,19000,20000
13215 IF A$="X" THEN BB=1 : GOTO 11110
13220 LOOP
13230 RETURN
```

Verifica

Eseguite il programma. Quando sarà stato disegnato il simbolo "@", dovrebbe essere sufficiente battere 0 per vedere comparire il menu. Dovreste poi poter spostare il cursore lampeggiante su tutto il quadratino senza danneggiarlo. Premendo 1, riempirete un nuovo pixel, premendo 0 lo potrete cancellare. A parte questi e il comando R, per il ritorno, tutti gli altri comandi non producono altro effetto che quello di interrompere il programma con un messaggio di errore.

Modulo 3.3.6: Creazione di un carattere in negativo

Iniziamo con questo una serie di moduli che facilitano la modifica (o l'editing, come si dice in gergo) del carattere, eseguendo operazioni sull'intero carattere, come, ad esempio, la creazione di un'immagine simmetrica o la rotazione del carattere di 90 gradi. Questo primo modulo crea semplicemente l'immagine in negativo di un carattere già esistente, annullando tutti i pixel attivati e attivando gli altri.

Modulo 3.3.6: Linee 15000-15070

```
15000 REM*******************************
15010 REM INVERSIONE
15020 REM*******************************
15030 FOR I=0 TO 7 : FOR J=0 TO 7
```

```
15040 LC=PEEK (3072+40*I+J)
15050 IF LC=81 THEN CHAR ,J,I," " : ELSE CHAR
,J,I,CHR$(113)
15060 NEXT J,I
15070 RETURN
```

Verifica

Eseguite il programma ed entrate nel modo edit. Non appena compare il menu, premete I e vedrete comparire il carattere trasformato nel suo corrispondente in negativo. Premete di nuovo I e il carattere ricomparirà nella sua forma originale. Quanto detto si riferisce esclusivamente ai caratteri ingranditi, non a quelli di dimensioni normali posti alla destra del quadrato 8*8.
Questi ultimi possono essere modificati solo se si decide di introdurre in memoria il carattere editato utilizzando il modulo che segue.

Modulo 3.3.7: Creazione di un'immagine speculare

Questo modulo prende il carattere visualizzato e lo trasforma nel suo corrispondente simmetrico, cioè come se fosse visto in uno specchio. A differenza del modulo precedente, che opera direttamente sullo schermo, questo memorizza il risultato, fino a trasformazione completa, nel vettore TT%. Questo perché, durante gli spostamenti all'interno del quadrato 8*8, il programma potrebbe andare a scrivere sopra il contenuto originale e sarebbe difficile, per non dire impossibile, distinguere fra ciò che è stato modificato e ciò che deve ancora esserlo.

Modulo 3.3.7: Linee 16000-16100

```
16000 REM*******************************
16010 REM RIFLESSO
16020 REM*******************************
16030 FOR I=0 TO 7 : FOR J=0 TO 7
16040 LC=PEEK (3072+40*I+J)
16050 IF LC=81 THEN TT%(I,J)=1 : ELSE
TT%(I,J)=0
16060 NEXT J,I
16070 PRINT "[HOME]"; : FOR I=0 TO 7 : FOR J=7
TO 0 STEP-1
16080 IF TT%(I,J) THEN PRINT CHR$(113); : ELSE
PRINT" ";
16090 NEXT J : PRINT : NEXT I
16100 RETURN
```

Commento

Linee 16030-16060: Il contenuto del quadrato 8*8 viene copiato direttamente nel vettore TT% sotto forma di insieme di uno a zeri.

Linee 16070-16090: Il contenuto di TT% viene ricopiato di nuovo sullo schermo, ma questa volta leggendo le linee da destra e stampandole da sinistra a destra (in altre parole, ogni linea viene stampata all'inverso).

Verifica

Eseguite il programma ed invocate il menu di editing. Premete M e, dopo una pausa, durante la quale il carattere viene copiato nel vettore, vedrete quest'ultimo riprodotto nella sua immagine simmetrica.

Modulo 3.3.8: Rotazione del carattere

Se provate ad immaginare il carattere che state editando come se fosse stampato su un foglio di plastica trasparente, tutto ciò che potete fare con esso può essere ottenuto combinando due interventi: la creazione di una sua immagine simmetrica e/o la sua rotazione di novanta gradi una o più volte. Anche questo modulo usa il vettore TT%, ma questa volta per ruotare il carattere nel quadrato 8*8 di 90 gradi in senso antiorario.

Modulo 3.3.8: Linee 17000-17100

```
17000 REM*******************************
17010 REM ROTAZIONE DEI CARATTERI
17020 REM*******************************
17030 FOR I=0 TO 7 : FOR J=0 TO 7
17040 LC=PEEK (3072+40*I+J)
17050 IF LC=81 THEN TT%(7-J,7-I)=1 : ELSE
TT%(7-J,7-I)=0
17060 NEXT J,I
17070 PRINT "[HOME]"; : FOR I=0 TO 7 : FOR J=7
TO 0 STEP-1
17080 IF TT%(I,J)=1 THEN PRINT CHR$(113); :
ELSE PRINT" ";
17090 NEXT J : PRINT : NEXT I
17100 RETURN
```

Commento

Linee 17070-17100: Questo ciclo copia di nuovo il contenuto del vettore in un quadrato. Questa volta, però, nel momento in cui il vettore viene letto, le variabili vengono invertite, in modo che il ciclo inizi dall'ultima colonna e scenda lungo di essa, stampando il contenuto in senso orizzontale sul bordo superiore del

quadrato. Poi continua con la seconda e avanti fino all'ultima colonna stampando i dati sulla seconda linea del quadrato e poi avanti fino all'ultima. Questo si traduce in una rotazione del carattere di 90 gradi in senso antiorario.

Verifica

Eseguite il programma, richiamate il menu di editing e poi premete il tasto T; dopo una breve pausa, vedrete comparire il carattere ruotato di novanta gradi. Premendo T altre tre volte, otterrete la comparsa dello stesso carattere nella sua posizione originale. Fate qualche esperimento per conto vostro, ruotando e invertendo vari caratteri finché non sarete soddisfatti.

Modulo 3.3.9: Memorizzazione di un carattere modificato

Finora siete stati in grado di manipolare il carattere ingrandito nel quadrato 8*8 fintantoché non era in relazione con il modello originale. Tutto questo, comunque, non comportava nessuna differenza per la versione di dimensioni normali del carattere che è riprodotto sulla destra del quadrato 8*8. I cambiamenti che avete imparato ad apportare e che avete effettivamente apportato non sono stati ancora introdotti nella memoria caratteri e non lo saranno finché ciò che avete ottenuto non vi soddisferà completamente. Una volta raggiunta il suo obiettivo, questo modulo trasforma la configurazione di pixel all'interno del quadrato in qualcosa che sarà, d'oro in avanti, parte del vostro set di caratteri personalizzato.

Modulo 3.3.9: Linee 18000-18080

```
18000 REM*******************************
18010 REM INSERIMENTO NEL SET DI MEMORIA
18020 REM*******************************
18030 FOR I=0 TO 7 : TT=0
18040 FOR J=0 TO 7 : LC=PEEK(3072+40*I+J)
18050 IF LC=81 THEN TT=TT+2↑(7-J)
18060 NEXT J
18070 POKE CP+I,TT : NEXT I
18080 RETURN
```

Commento

Linea 18050: Questa è l'unica linea che fa' qualcosa di realmente nuovo, tutte le altre analizzano la configurazione come nel modulo precedente. Il suo effetto è quello di prendere una posizione in cui c'è un cerchio verde e di tradurla nella cifra binaria di un numero che rappresenta una riga di otto spazi. Ogni cerchio rappresenta una potenza di due usata nel modo che abbiamo visto analizzando la possibilità di usare i numeri binari per formare le singole linee orizzontali che costituiscono il carattere. Quando tutte le otto linee sono state tradotte in un

numero, vengono scritte (con POKE) nella memoria caratteri nella posizione corrispondente ad una delle linee orizzontali del carattere elaborato al momento.

Verifica

Eseguite il programma e spostate il puntatore dei caratteri sul carattere uno, cioè A. Trasferitevi nel modo di editing e fate ruotare il carattere una volta. Ora premete il tasto P e osservate lo schermo. Come noterete, non solo viene trasformata la A a destra del quadrato 8*8, ma tutte le A sullo schermo risultano girate di lato. Nel darvi questo effetto, il C16 usa le informazioni ricavate dal vostro set personalizzato. Ricordate che, nell'editare i caratteri, è meglio scegliere quelli grafici. Troppe modifiche a lettere e numeri possono dar luogo ad una situazione in cui non si capisce più che cosa vuole comunicare il programma!

Modulo 3.3.10: Memorizzazione del set di caratteri

Dopo aver modificato il set di caratteri, vogliamo ora poterlo conservare, in modo da usarlo in seguito per altri scopi (se ciò non fosse possibile, il programma perderebbe di utilità). Questo modulo serve per l'appunto a memorizzare su disco il set di caratteri modificato secondo i desideri dell'utente. Non ci sono grosse sorprese, come vedrete: il modulo usa esattamente lo stesso metodo impiegato nel programma ARTIST per salvare l'immagine su schermo. L'unica differenza è data dal fatto che, per rappresentare il luogo in cui viene memorizzato il set di caratteri, le istruzioni POKE definiscono un'area di memoria da 14336 a 16384.

Modulo 3.3.10: Linee 19000-19070

```
19000 REM*****************************
19010 REM SALVA UN SET DI CARATTERI
19020 REM*****************************
19030 FOR I=0 TO 3 : A%(I)=PEEK(43+I) : NEXT
19040 POKE 43,0 : POKE 44,14*4 : POKE45,0 :
POKE 46,16*4
19050 SAVE "@0:CHARDATA",8
19060 POKE 43,A%(0) : POKE 44,A%(1) : POKE
45,A%(2) : POKE 46,A%(3)
19070 RETURN
```

Verifica

Dopo aver editato un po' di caratteri e averli trasferiti in memoria, salvateli su disco utilizzando questo modulo. La sola conferma del buon funzionamento del modulo, a questo punto è data dal fatto che può essere eseguito senza produrre messaggi di errore. Dopo che sarà stato introdotto anche il prossimo modulo, potrete ricaricare tutti i caratteri e controllare che siano stati memorizzati correttamente.

Modulo 3.3.11: Ricamento del set di caratteri

Una volta avvenuta la memorizzazione del set di caratteri su disco, i dati relativi possono essere richiamati o ricaricati usando questo modulo. Lo sola differenza fra questo e il metodo usato nel programma ARTIST, per ricaricare un'immagine su schermo, è che mentre ha luogo il caricamento, il C16 salta temporaneamente al set di caratteri conservato su ROM. Come per il programma ARTIST, il programma viene eseguito automaticamente non appena il set è stato caricato.

Modulo 3.3.11: Linee 20000-20060

```
20000 REM*******************************
20010 REM CARICA UN SET DI CARATTERI
20020 REM*******************************
20030 TT=DEC("FF12")
20040 POKE TT+1,208
20050 POKE TT,PEEK(TT) OR 4
20060 LOAD "CHARDATA",8,1
```

Verifica

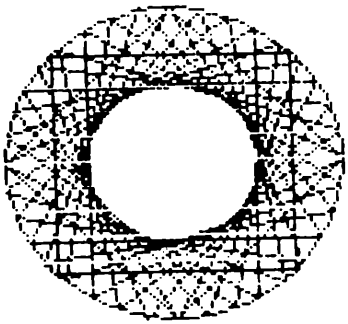
A questo punto dovreste essere in grado di caricare il set di caratteri che avete salvato (SAVE) come parte della prova per il modulo precedente, premendo L nel modo di editing dei caratteri. Prima di caricare quello che avete appena salvato, assicuratevi di essere tornati al set dei caratteri normali (non modificati) o sarà impossibile stabilire se sono stati caricati da disco dei caratteri diversi.

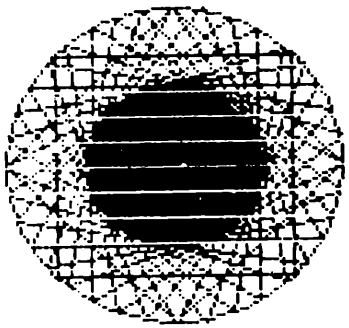
Programma 3.4: HDRAW

Proviamo ora ad esaminare alcune delle possibilità più creative e divertenti della grafica ad alta risoluzione. Se date un'occhiata alla lunghezza del programma ARTIST, capirete immediatamente che con 8K di memoria usati per lo schermo ad alta risoluzione, non avrete mai abbastanza spazio per poter creare qualcosa di sofisticato.

Questo non significa comunque che non possiate realizzare qualcosa di utile e divertente. Il programma che state per conoscere è stato studiato per presentarvi alcuni dei comandi di gestione della grafica ad alta risoluzione e per darvi la possibilità di divertirvi con il vostro computer. Con il nostro programma HDRAW, potrete trasformare in realtà le immagini della vostra fantasia e creare disegni formati da linee, quadri, cerchi ed altre figure geometriche elementari, usando un piccolo cursore lampeggiante per spostarvi dove volete che gli elementi del disegno vadano a collocarsi. Con un po' di pratica, oltre a permettervi di creare disegni interessanti e piacevoli, il programma vi darà gli strumenti adatti ad approfondire la conoscenza dei comandi grafici.

Figura 3.2: Semplice disegno realizzato in due minuti con il programma HDRAW





Nel corso del programma, parleremo di due cose nuove:

1) L'uso interattivo dei comandi grafici
2) L'uso della grafica a schermo ripartito

Modulo 3.4.1: Inizializzazione

Quello che riportiamo qui in basso è un modulo di inizializzazione molto normale. Per risparmiare spazio, abbiamo tralasciato le linee di asterischi e abbiamo messo i titoli dei moduli fra parentesi. Per mantenere l'uniformità con gli altri programmi del libro, però, la parte funzionante del modulo inizia ancora a partire dalla linea ***30 delle migliaia significative (in altre parole non manca nessuna linea).

Modulo 3.4.1: Linee 10000-10100

```
10000 REM INIZIALIZZAZIONE**************
10020 COLOR 0,2 : COLOR 1,1
10030 TRAP 10080
10040 X=160 : Y=100 : DC=1
10050 MENU$="12LBCP[HOME]*U"
10060 GRAPHIC 2,1-IN : IN=1
10070 GOSUB 11000
10080 GRAPHIC 0,1
10090 PRINT ERR$(ER),EL
10100 END
```

Modulo 3.4.2: Subroutine ad una linea

A causa dello spazio molto limitato, abbiamo deciso di essere molto contenuti nell'uso dell'istruzione ON...GOSUB (ogni funzione è inserita in un MENU$ invece

di essere accessibile per mezzo di un costrutto IF...THEN sulla base di quanto viene battuto dall'utente). Questa scelta rende necessaria una serie di subroutine ad una linea che dovrebbero essere altrimenti inserite nel modulo di controllo. Queste linee svolgono una serie nutrita di funzioni che vedremo meglio nel commento al modulo di controllo.

Modulo 3.4.2: Linee 16000-16090

```
16000 REM SUBROUTINE SINGOLE************
16030
PRINT"[HOME][CD][CD][CD][CD][CD][CD][CD][CD][C
D][CD][CD][CD][CD][CD][CD][CD][CD][CD][CD][CD]
"; : RETURN
16040 X1=X : Y1=Y : RETURN
16050 X2=X : Y2=Y : RETURN
16060 DRAW DC,X1,Y1 TO X2,Y2 : RETURN
16070 DRAW 0,X,Y : PAINT ,X,Y : RETURN
16080 SCNCLR : RETURN
16090 DC=1-DC : RETURN
```

Modulo 3.4.3: Il modulo di controllo

Come sempre, questo modulo crea un collegamento fra le diverse parti del programma. I soli compiti riservati al modulo di controllo sono la modifica delle coordinate del cursore e l'analisi del MENU$ per vedere se l'utente ha battuto un comando valido.

Modulo 3.4.3: Linee 11000-11120

```
11000 REM MODULO DI CONTROLLO***********
11030 DO
11040 GOSUB 15000
11050
X=X-(A$="[CR]")+(A$="[CL]")+10*((A$="F")-(A$="
G"))
11060
Y=Y-(A$="[CD]")+(A$="[CU]")+10*((A$="T")-(A$="
V"))
11070 X=X+319*((X>319)-(X<0))
11080 Y=Y+159*((Y>159)-(Y<0))
11090 Z=INSTR(MENU$,A$)
11100 ON Z GOSUB
16040,16050,16060,12000,13000,16070,16080,1609
```

```
0,14000
11110 PRINT "[CLEAR]"
11120 LOOP
```

Commento

Linee 11050-11080: Abbiamo già parlato dell'uso delle condizioni logiche come valori. In queste due linee, la tecnica viene usata per eliminare almeno 12 istruzioni IF. Il loro effetto è quello di modificare le coordinate del cursore di un pixel ogni volta che vengono usati i tasti di controllo cursore e di 10 pixel quando vengono invece usati i tasti

```
  T
F   G
  V
```

Se il cursore esce dal bordo dello schermo, ricompare sempre sul lato opposto. Poiché stiamo usando il modo grafico a schermo ripartito, dove le ultime cinque linee in basso sono destinate al testo, la posizione più in basso che il cursore può raggiungere è la posizione 159, e non la 199 come nello schermo interamente dedicato alla grafica.

Linee 11090-11100: I comandi accettati da queste linee, sulla base del contenuto di MENU$, sono:
1 — Definisce l'inizio della linea da tracciare o un angolo del quadrato che dovrà contenere una figura ancora da specificare. Le coordinate del punto sono memorizzate nelle variabili X1 e Y1 dalla linea 16040.
2 — Definisce la fine della linea da disegnare o l'angolo opposto del quadrato. Le coordinate vengono memorizzate nelle variabili X2 e Y2 dalla linea 16050.
L — Fra i punti definiti premendo 1 e 2 viene tracciata una linea nel colore corrente (DC), che può essere sia di primo piano che di sfondo (vedi comando *).
B — Viene disegnato un quadrato usando i punti uno e due come angoli opposti. Il disegno del quadrato viene eseguito dal modulo alla linea 12000.
C — Viene disegnato un cerchio in modo tale che vada a toccare i lati del quadrato ipotetico creato dai due punti. Il disegno del cerchio viene eseguito dal modulo alla linea 13000.
P — Il comando PAINT viene chiamato dalla posizione corrente del cursore. L'effetto è ottenuto con la linea 16070.
[HOME] — Il tasto HOME/CLR azzera lo schermo grafico (non occorre usare il tasto SHIFT). L'operazione è eseguita dalla linea 16080.
* — Tutte le figure e tutte le linee possono essere disegnate sia nel colore di fondo che in quello di traccia. Il colore di fondo viene normalmente usato per cancellare ciò che già esiste sullo schermo. Ogni volta che si preme il tasto asterisco, il colore visibile si trasforma da colore di fondo a colore di traccia o di

primo piano e viceversa. L'operazione è eseguita dalla linea 16090.
U — Premendo U si possono ottenere informazioni sullo stato attuale delle principali variabili da stampare, compresa la posizione del cursore, la posizione definita per i punti uno e due e la larghezza e la profondità del riquadro definito.

Verifica

Con i primi tre moduli introdotti è possibile eseguire una verifica di un certo significato. Battete la linea provvisoria:

```
15000 RETURN
```

ed eseguite il programma. Lo schermo passerà nel modo ad alta risoluzione a schermo ripartito.

Modulo 3.4.4: Il cursore lampeggiante

La sola differenza fra questo modulo e quello che già conoscete per il modo a bassa risoluzione è che qui lampeggia un solo pixel.

Modulo 3.4.4: linee 15000-15110

```
15000 REM*******************************
15030 LOCATE X,Y
15040 SR=RDOT(2)
15050 A$="" : DO UNTIL A$<>""
15060 DRAW 1-SR,X,Y
15070 FOR I=1 TO 50 : NEXT
15080 DRAW SR,X,Y
15090 FOR I= 1 TO 50 : NEXT
15100 GET A$ : LOOP
15110 RETURN
```

Commento

Linee 15030-15040: Prima di far diventare il cursore lampeggiante, viene ricavato il colore (del disegno o di fondo) della sua attuale posizione.

Linee 15050-15100: Viene stampato un unico pixel, prima del colore opposto a quello originale, poi dello stesso colore (non appena il ciclo ha termine, il pixel viene ripristinato nel suo stato originale).

Verifica

Il programma dovrebbe comportarsi come nell'ultima prova, con la differenza però che ora dovreste essere in grado di vedere un piccolo cursore lampeggiante. Con il programma in esecuzione e con il cursore lampeggiante, battete 1,G,G,2,L

e dovreste veder comparire una breve linea. Per cancellarla premete *,L, il comando che modifica il colore in quello di fondo e ridisegna la stessa linea.

Modulo 3.4.5: Aggiornamento dei dati

Questo modulo stampa le informazioni più importanti sullo stato corrente del programma sulla linea più bassa dello schermo.

Modulo 3.4.5: Linee 14000-14080

```
14000 REM INFORMAZIONI ****************
14030 GOSUB 16030 : PRINT "X =";X;"      Y =";Y
14040 PRINT "X1=";X1;"        Y1=";Y1
14050 PRINT "X2=";X2;"        Y2=";Y2
14060 PRINT "LARGHEZZA=";X2-X1;"
PROFONDITA'=";Y2-Y1
14070 PRINT"PREMI UN TASTO"; : GETKEY Q$
14080 RETURN
```

Commento

Linea 16030: Questa GOSUB invoca una linea che sposta la posizione di stampa sulla ventunesima linea dello schermo destinato al testo, dove può essere vista nel modo di schermo ripartito. Purtroppo per questo non è possibile usare l'istruzione CHAR dato che, con il modo grafico due attivato, CHAR,0,20 insiste con la stampa su una parte dello schermo grafico che non è disponibile, invece che nello spazio dedicato al testo.

Verifica

Il programma dovrebbe essere in grado di funzionare come prima con la sola differenza che ora, premendo U, si avrà un aggiornamento sulla posizione corrente del cursore e sulla definizione delle posizioni 1 e 2.

Modulo 3.4.6: Creazione di un quadrato

In questo modulo impareremo a definire due angoli e ad usare questa capacità per disegnare un quadrato (due angoli sono più che sufficienti). I quadrati vengono di fatto creati usando il comando CIRCLE, che vedremo più da vicino in seguito, il quale però obbliga a definire quattro paremetri. Una volta che gli angoli sono stati definiti, occorre comunicare al comando BOX di quanto deve essere fatta ruotare la figura intorno al punto posto a metà fra i due angoli.

Modulo 3.4.6: Linee 12000-12060

```
12000 REM BOX ***************************
12030 RO=0 : GOSUB 16030 : INPUT "ROTAZIONE
(0-360):";RO
```

```
12040 BOX DC,X1,Y1,X2,Y2,RO
12050 RETURN
```

Verifica

Battete il comando RUN, per eseguire il programma, poi 1 per definire un angolo. Spostate quindi il cursore prima a destra e poi in basso e premete il 2, seguito da B e, quando vi viene chiesto l'angolo di rotazione, premete il tasto RETURN. Dovreste veder disegnarsi un quadrato con due angoli nelle posizioni occupate dal cursore nel momento in cui sono stati premuti i tasti 1 e 2. Ricordate che non importa dove spostate il cursore dopo che avete definito gli angoli: il quadrato cambia solo se introducete una nuova definizione di angolo. Fate qualche esperimento con diverse rotazioni degli stessi angoli e osservate l'effetto di questo parametro. I quadrati disegnati sullo schermo possono essere cancellati cambiando il colore dominante con il comando asterisco "*".

Modulo 3.4.7: Uso del comando CIRCLE

Già nella presentazione dell'ultimo comando abbiamo osservato che CIRCLE è molto più complesso del comando BOX. Il modulo che vi presentiamo qui vi consente di usarne tutta la potenza, pur costringendovi a introdurre i quattro parametri di cui il comando ha bisogno.

Modulo 3.4.7: Linee 13000-13110

```
13000 REM CERCHIO ***********************
13030 CX=X1+INT((X2-X1)/2)
13040 CY=Y1+INT((Y2-Y1)/2)
13050 SA=0 : GOSUB 16030 : INPUT "INIZIO
ANGOLO (0-360):";SA
13060 EA=360 : INPUT "FINE ANGOLO (0-360):";EA
13070 RO=0 : INPUT "ROTAZIONE (0-360):";RO
13080 DEG=2 : INPUT "GRADI FRA I
SEGMENTI:";DEG
13090 CIRCLE
DC,CX,CY,ABS(CX-X1),ABS(CY-Y1),SA,EA,RO,DEG
13100 RETURN
```

Commento

Linee 13030-13040: Le due linee definiscono le coordinate del centro del cerchio che verrà disegnato.

13050-13080: Queste linee permettono di definire i parametri richiesti dal comando che esegue la traccia del cerchio. Premendo RETURN in risposta a qualsiasi

richiesta del programma, il parametro acquisito è quello predefinito (di default). Se la funzione dei parametri non vi è molto chiara, o se non sapete che cos'è un valore di default, consultare il manuale mentre vi cimentate con questo modulo (un esempio concreto vale più di mille parole).

Linea 13090: Disegnando sulle posizioni definite per gli angoli 1 e 2, questa linea inserisce altri parametri, non fissati direttamente da voi, e disegna il cerchio specificato o qualsiasi altra figura geometrica.

Verifica

Eseguite il programma, definite due angoli distinti e premete il tasto C. Premete il tasto RETURN in risposta a tutti i solleciti del programma e vedrete il cerchio disegnarsi sullo schermo. Premete quindi B e poi RETURN non appena vi viene chiesto di specificare la rotazione e, intorno al cerchio, comincerà a disegnarsi un quadrato. Tutto questo per dimostrarvi in che modo il programma HDRAW acquisisce le dimensioni comunicate con il comando E (ogni cerchio disegnato tocca sempre i centri di tutti i lati del quadrato che può essere disegnato usando le stesse definizioni di angolo. Questo non è sempre ovvio dato che se aumentate il valore introdotto in risposta al quarto sollecito "gradi fra i segmenti", in modo da produrre figure geometriche a profilo netto e regolare, queste figure vengono disegnate dentro la linea del cerchio immaginario con i soli angoli a contatto.

Per verificarlo, senza dover ridefinire uno dei due angoli, premete di nuovo il tasto C. Poi premete RETURN a tutti i solleciti tranne il quarto per il quale battete il numero 120. Vedrete disegnarsi all'interno del cerchio un triangolo, di cui solo l'angolo più alto risulta a contatto con il quadrato.

Programma 3.5: MUSIC

Nessun libro sul C16 può dirsi completo senza un adeguato discorso sulla flessibilità d'impiego dei comandi di controllo del suono, che, a differenza dei suoi predecessori, la nuova macchina è in grado di offrire. Nel programma che stiamo per illustrarvi, esamineremo un metodo per usare appieno i comandi SOUND e VOL per creare piccoli brani musicali, senza dover fare ricorso ad un lungo elenco di comandi o alle migliori doti, per scoprire il giusto valore di ogni nota. Facendo uso delle istruzioni DATA, il programma permette di introdurre brani di una certa lunghezza, a una o due voci, in un formato lineare, con controllo del volume, della durate della nota e dell'ottava.

Gli argomenti completamente nuovi, affrontati in questo programma sono:

1) L'uso più sofisticato del comando SOUND
2) L'uso dei vettori multidimensionali
3) L'aumento della velocità di esecuzione mediante la preelaborazione dei dati.

Modulo 3.5.1: Inizializzazione

Modulo 3.5.1: Linee 10000-10040

```
10000 REM*******************************
10010 REM INIZIALIZZAZIONE
10020 REM*******************************
10030 DIM PLAY%(200,1,2)
10040 FR=110.0875
10050 OC=2 : LO=4
```

Commento

Linea 10030: Per capire questa linea, dovete tornare indietro alle lezioni di matematica e cercare di ricordare le nozioni apprese a scuola sul concetto di dimensione. Infatti, il tipo speciale di variabili, note con il nome di "vettori", si comportano quasi nello stesso modo delle linee, delle superfici e dei solidi che hanno rispettivamente una, due e tre dimensioni.

1) Vettori unidixensionali. Li abbiamo usati spesso. Sia di stringa che numerici, essi non sono altro che delle liste, in modo che, per esempio:

A$(100)

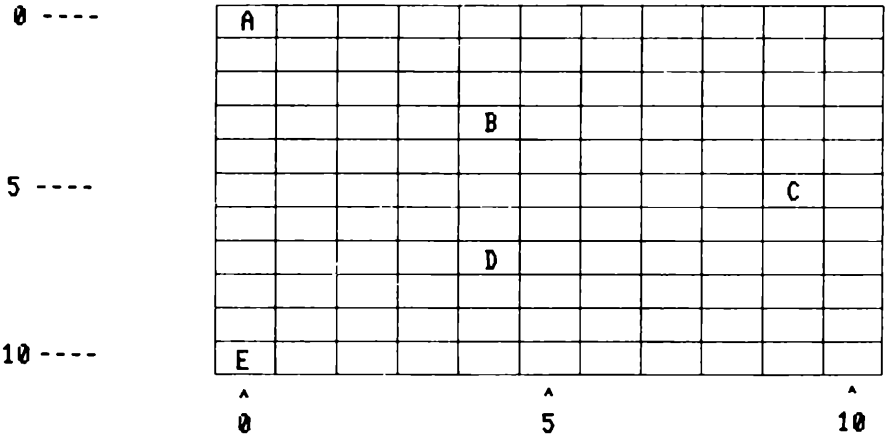
equivale ad una linea che può essere divisa in 101 (non dimenticate lo zero!) unità:

```
---------------------.........---------------------
^                   ^                ^                ^
0                   20               80               101
```

2) Vettori bidimensionali. Un vettore come:

A$(10,10)

può essere visto come un rettangolo, contenente spazio sufficiente per 11*11 elementi:



```
0 ----   | A |   |   |   |   |   |   |   |   |   |   |
         |   |   |   |   |   |   |   |   |   |   |   |
         |   |   |   |   |   |   |   |   |   |   |   |
         |   |   |   |   | B |   |   |   |   |   |   |
         |   |   |   |   |   |   |   |   |   |   |   |
5 ----   |   |   |   |   |   |   |   |   |   | C |   |
         |   |   |   |   |   |   |   |   |   |   |   |
         |   |   |   |   | D |   |   |   |   |   |   |
         |   |   |   |   |   |   |   |   |   |   |   |
         |   |   |   |   |   |   |   |   |   |   |   |
10 ----  | E |   |   |   |   |   |   |   |   |   |   |
         ^                       ^                       ^
         0                       5                       10
```

In questo caso, le posizioni delle celle sono le seguenti:

A — 0,0
B — 4,3
C — 9,5
D — 4,7
E — 0,10

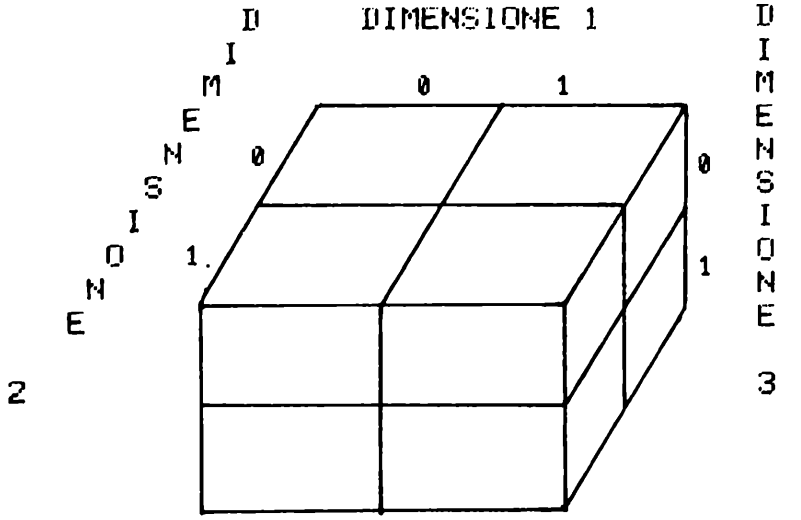
In effetti è più che plausibile vedere un elenco a due dimensioni come una serie di liste bidimensionali:

LIST 2:

A										

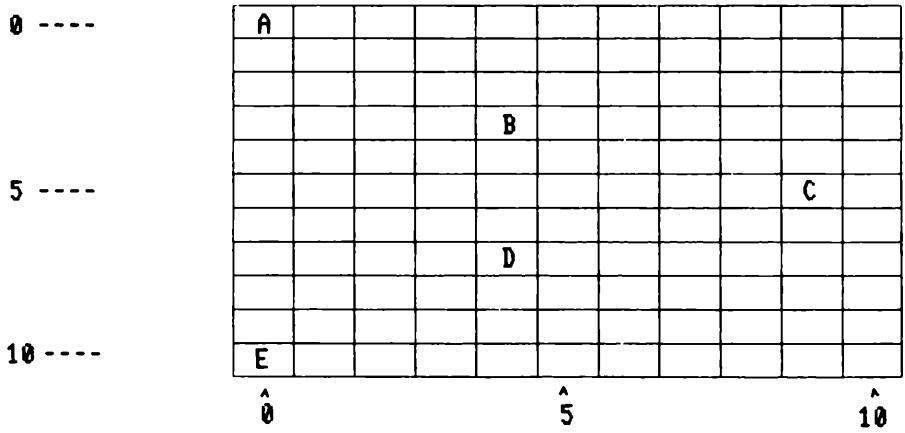
3) Vettori tridimensionali. Se la maggior parte della gente sembra non avere alcuna difficoltà nel capire i vettori uni- e bi-dimensionali, va letteralmente in confusione quando, nei programmi, si comincia a parlare di vettori con tre o più dimensioni. Questo programma usa proprio un vettore tridimensionale, che può essere visto come un parallelepipedo suddiviso in celle, ma potrebbe essere anche un certo numero di elenchi identici a due dimensioni:

a) Vettore trimendisionale visto come un parallelepipedo

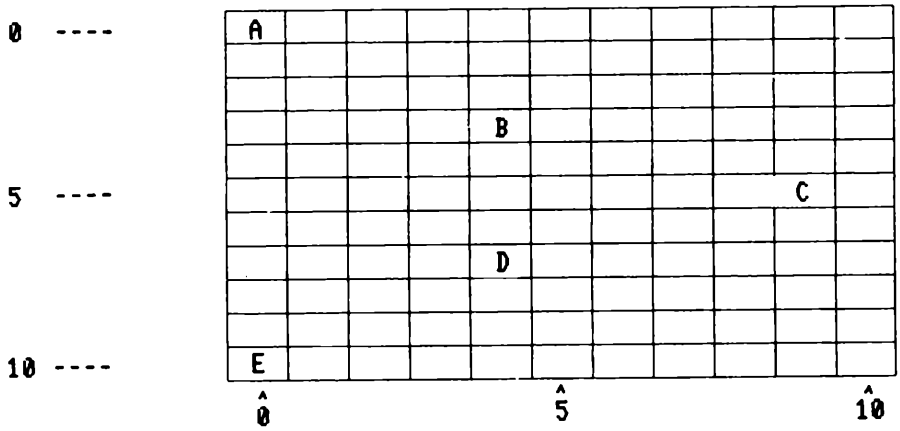


b) Vettore tridimensionale visto come una serie di vettori bidimensionali identici

i) griglia anteriore



ii) griglia inferiore



qui, la cella A della griglia anteriore si troverà nella posizione 0,0,0, mentre la cella D della griglia posteriore si troverà nella posizione 4,7,1.
Tutto quanto detto serve solo ad aiutarvi a capire concetti che possono confonder-

vi quando dovrete introdurre e cercare di capire queste linee che fanno uso del vettore PLAY%. Essenzialmente, PLAY% è una coppia di liste di 201 note al massimo, destinate ad essere suonate in ciascuna delle due voci del C16 (non useremo la terza). Ogni elemento delle due liste ha tre celle che verranno usate per memorizzare i dati relativi al volume, al valore della nota e alla sua durata. In altri termini, ciò che abbiamo è una coppia di liste in questa forma:

	LISTA PER LA VOCE 1	LISTA PER LA VOCE 2
LINEA 1	Voce 1, nota 1 (3 dati)	Voce 2, nota 1 (3 dati)
-		
-		
-		
LINEA 201	Voce 1, nota 201 (3 dati)	Voce 2, nota 201 (3 dati)

Tutte le linee che si riferiscono, ad esempio, a PLAY%(20,....) avranno quindi qualcosa a che fare con la ventunesima nota della lista per una delle voci. Un riferimento a PLAY% (...,0,....) significerà l'indicazione della prima voce. Infine, la terza cifra fra parentesi indicherà se ciò cui si fa' riferimento è il volume, il valore nella nota o la sua durata (0, 1, o 2).

Tutto questo può sembrare a prima vista molto complicato e, a dire il vero, sarebbe possibile memorizzare tutti i dati di un suono in un vettore unidimensionale come PLAY%(1000). Ogni volta che volessimo ricavare i dati di una certa nota, però, dovremmo eseguire un bel po' di calcoli per scoprire dove si trovano, provocando un rallentamento del programma e aumentandone in definitiva la complessità. Il vantaggio della struttura tridimensionale di PLAY% è che si adatta ai dati come un guanto. Una volta che avete capito che cos'è quella cosa numerata lungo una delle tre dimensioni, potete ottenere tutto ciò che vi pare, semplicemente descrivendolo in questa forma, cioè ad esempio: "Voglio il terzo elemento, per la quarta nota sulla voce 2" e, nella forma corretta, "Voglio PLAY%(3,1,2), ricordando sempre che prima dell'uno c'è l'elemento zero.

Linea 10040: La frequenza di una nota di A in cicli al secondo. (La tabella con il confronto dei valori che devono essere usati nel comando SOUND, con le frequenze effettive prodotte, è riportata nel vostro manuale. Comunque, il valore assegnato ad A nel manuale è 110. Il valore più preciso dato in questa linea ci permetterà di stabilire i numeri relativi alle altre note mediante opportuni calcoli invece che utilizzando la tabella in memoria che fornisce le diverse frequenze.

Modulo 3.5.2: I dati musicali

Come abbiamo già detto nell'introduzione, lo scopo di questo programma è di permettervi di introdurre dei dati musicali in un modo più comprensibile di quello

consistente nell'usare lunghi elenchi di numeri complicati.
Il sistema adottato usa numeri, ma solo quelli compresi fra 1 e 12 (inclusi), cioè tutti i semi-toni di una normale ottava musicale. Questo non significa comunque che siate limitati all'ottava: esiste infatti una funzione di definizione dell'ottava, dove l'ottava uno è la più bassa e l'ottava sei la più alta in termini di utilità musicale. Oltre a specificare il tono della nota, è possibile stabilire quanto deve essere lunga e il livello del volume ad un certo punto del motivo musicale.
Le informazioni vengono memorizzate nelle istruzioni DATA in questa forma:
Valore della nota: 1-12, con 1 che rappresenta A
Ottava: O, seguito da 1-6, cambia l'ottava
Durata: L seguito da qualsiasi numero valido in un comando di controllo del suono. L'uso delle potenze di due può semplificare la traduzione della notazione musicale.
Volume: V, seguito da un numero da 0 a 8. Il volume ha effetto su entrambe le voci e il valore zero spegne entrambe (anche se comunque le note continuano ad essere prodotte in silenzio).
Tutte le voci sono separate da virgole e il salto su una nuova linea di DATA non influenza minimamente il motivo. Le istruzioni DATA devono terminare con un'istruzione dello stesso tipo che legge END. Se volete usare una sola voce, l'istruzione DATA per la voce uno deve essere seguita da due END consecutivi. L'istruzione DATA di questo modulo usa entrambe le voci. La voce 1 produce le note basse, la voce 2 suona più rapidamente e su toni più alti.

Modulo 3.5.2: Linee 14000-14070

```
14000 REM*******************************
14010 REM DATA
14020 REM*******************************
14030 DATA V3,L32,O2,1,3,5,6,8,10,12,O3,1
14040 DATA FINE
14050 DATA V3,L16,O5,1,3,5,6,8,10,12,O6,1
14060 DATA V3,L16,O3,1,3,5,6,8,10,12,O4,1
14070 DATA FINE
14150 SOUND
2,PLAY%(NN(1),1,1),PLAY%(NN(1),1,2)
```

Modulo 3.5.3: Estrazione di un valore da un dato

Poiché le informazioni su cui si deve basare il motivo musicale non vengono introdotte in una forma che potrebbe essere usata direttamente in un comando SOUND, ad un certo punto devono essere tradotte. Questo modulo esegue per l'appunto quest'operazione per un elemento di un'istruzione DATA.

Modulo 3.5.3: Linee 12000-12140

```
12000 REM*********************************
12010 REM ANALISI STRINGA MUSICALE
12020 REM*********************************
12030 TP$="" : DO UNTIL VAL(TP$)<>0
12040 READ TP$
12050 IF TP$="FINE" THEN RETURN
12060 IF LEFT$(TP$,1)="V" THEN
LO=VAL(MID$(TP$,2))
12070 IF LEFT$(TP$,1)="O" THEN
OC=VAL(MID$(TP$,2))
12080 IF LEFT$(TP$,1)="L" THEN
NL=VAL(MID$(TP$,2))
12090 LOOP
12100 DO UNTIL VAL(TP$)=0
12110 NO=12*(OC-1)+VAL(TP$)
12120 NO=1024-(111840.45/(FR*1.05946309↑NO))
12130 EXIT : LOOP
12140 RETURN
```

Commento

Linee 12030-12090: Questo ciclo prende un elemento dell'istruzione DATA e fa' una di queste tre cose:
a) un RETURN se la stringa è END
b) niente se l'elemento è un valore di nota
c) memorizza il valore in una variabile se l'elemento è un'istruzione per la modifica del volume, dell'ottava o della durata e continua a cercare un valore di nota o un END.

Linee 12100-12130: Se l'istruzione DATA non inizia con una lettera (V,O o L), deve essere un valore di nota. Questo ciclo traduce quel valore in due fasi distinte:
i) tenendo conto dell'ottava corrente (linea 12110) e di quando viene ottenuta;
ii) calcolando il numero che dovrebbe essere usato in un comando SOUND per produrre la nota specificata. I calcoli si dividono in due parti. La prima è costituita dalla formula specificata nell'appendice del manuale dell'utente, che contiene la tabella delle note musicali e che dipende dal fatto che il modello di C16 utilizzato sia o meno predisposto per l'uscita su un televisore conforme alle norme americane NTSC o su un televisore PAL. La seconda è invece costituita dalla formula che ricava la frequenza moltiplicando la frequenza della nota A (vedi modulo 3.5.1) per 1,05946309 elevato alla potenza del valore di nota. In altre parole,

questa parte del calcolo si basa sull'assunto che la frequenza di qualsiasi semitono può essere ottenuta moltiplicando la frequenza del semi-tono successivo più basso per 1,05946309. Per ottenere, ad esempio, la frequenza della nota 20, che è l'ottava nota dell'ottava due, non dovete fare altro che moltiplicare la frequenza della A più bassa (110,0875) per 1,05946309, 20 volte, cioè 1,05946309^20.

Verifica

Date il via con il comando RUN all'esecuzvone del programma che genererà quasi subito il messaggio di errore RETURN WITHOUT GOSUB. Battete quindi quanto segue:

```
?LO[RETURN]
?OC[RETURN]
?NL[RETURN]
?NO[RETURN]
```

I risultati che dovete ottenere sono 3,2,32 e 544,548052 (ma l'ultima cifra sarà diversa se usate un televisore NTSC). Questo per ora è tutto quello che potete verificare.

Modulo 3.5.4: Controllo

Questo è il modulo che controlla la traduzione dei dati musicali e la loro successiva riproduzione. Una cosa interessante qui è che i valori ottenuti nel modulo precedente non vengono ancora usati nel comando SOUND. Questo perché si perderebbe molto tempo ad eseguire tutti i calcoli complicati che sono necessari per entrambe le voci, soprattutto se si pensa alla velocità necessaria per eseguire il motivo musicale. Non esistono praticamente microcomputer (salvo quelli che hanno un comando PLAY speciale per la produzione di sequenze di note) che siano tanto veloci da tradurre i dati e da suonare nello stesso tempo un motivo accettabile.
Se provate, compaiono fra le note lunghi intervalli, prodotti dal tempo necessario per eseguire tutti i calcoli, mentre le due voci risultano mal sincronizzate.
Se volete ottenere un rapido risultato sulla base di dati che come qui devono essere in qualche modo tradotti, la soluzione migliore è quella di elaborare i dati, memorizzarli nella loro forma definitiva e solo a questo punto usarli. Nel caso di questi programmi, il vettore PLAY% conterrà tutti i valori necessari per eseguire il motivo prima che venga suonato.

Modulo 3.5.4: Linee 11000-11120

```
11000 REM*******************************
11010 REM MODULO CONTROLLO MUSICA
11020 REM*******************************
11030 FOR VO=0 TO 1 : NN=0
```

```
11040 DO
11050 GOSUB 12000 : IF TP$="FINE" THEN EXIT
11060 PLAY%(NN,VO,0)=LO : PLAY%(NN,VO,1)=NO :
PLAY%(NN,VO,2)=NL
11070 NN=NN+1 : LOOP
11080 PLAY%(NN,VO,0)=-1
11090 NEXT VO
11100 VOL 2
11110 GOSUB 13000
11120 FOR I=1 TO 500 : NEXT : VOL 0 : STOP
```

Commento

Linee 11030 e 11090: Questo ciclo elabora ogni serie di istruzioni DATA per le due voci separatamente (tutti i dati relativi alla voce uno e poi quelli relativi alla voce 2).

Linee 11040-11070: Viene invocato il modulo precedente e viene ricavato il valore successivo dall'istruzione DATA. Abbiamo già visto che l'esecuzione ritorna sempre dal modulo precedente con un nuovo valore di nota, indipendentemente dal fatto che sia stato incontrato uno specificatore V, O o L. Ogni volta che l'esecuzione ritorna a questo modulo, le cifre per il volume, il valore della nota e la sua durata vengono memorizzate nel vettore PLAY% nella posizione corrispondente al numero della nota nell'ordine per quella voce (NN) e al numero della voce stessa (VO).

Linea 11080: Quando sono stati memorizzati tutti i dati di ciascuna voce, il primo di quelli che avrebbero dovuto essere i tre valori per la nota immediatamente successiva viene posto a meno uno. Questo servirà in seguito ad indicare che il motivo è terminato, per quanto riguarda quella voce.

Linee 11100-11120: Queste linee richiamano il modulo successivo che suona il motivo. Al ritorno da questo punto, ci può essere una nota che continua a suonare. Invece di interromperla bruscamente abbassando completamente il volume con VOL posto a zero, abbiamo preferito inserire un ciclo di temporizzazione che permette alla nota di terminare naturalmente.

Verifica

Il solo modo davvero efficace per provare il corretto funzionamento di un modulo, almeno a questo punto, e di usare un modulo provvisorio come quello riportato qui di seguito. Volendo, potreste anche aspettare fino a quando non avete introdotto il modulo PLAY, ma, nel caso in cui scopriste degli errori, dovreste comunque introdurre il modulo di prova. Quindi, tanto vale. Il suo scopo è di riprodurre su schermo i valori memorizzati per le voci uno e due.

Modulo provvisorio di prova

```
50000 REM*******************************
50010 REM TEST
50020 REM*******************************
50025 OPEN 1,4
50030 FOR I=0 TO 18 : FOR J=0 TO 1 : FOR K=0
TO 2
50040 PRINT#1,PLAY%(I,J,K); " ";
50050 NEXT K : IF J=0 THEN PRINT#1,"--";
50060 NEXT J : PRINT#1 : NEXT I
50070 CLOSE 1
```

Battete la linea provvisoria:

11105 STOP

ed eseguite il programma. Dopo una pausa abbastanza lunga, vedrete comparire il messaggio di errore BREAK. Battete:

GOTO 50000[RETURN]

e sullo schermo comparirà la Tabella 3.5.1.

Tabella 3.5.1

```
 3   544   32  -- 3   964   16
 3   596   32  -- 3   970   16
 3   643   32  -- 3   976   16
 3   664   32  -- 3   979   16
 3   704   32  -- 3   984   16
 3   738   32  -- 3   988   16
 3   770   32  -- 3   992   16
 3   784   32  -- 3   994   16
-1   0   0  -- 3   784   16
 0   0   0  -- 3   810   16
 0   0   0  -- 3   833   16
 0   0   0  -- 3   844   16
 0   0   0  -- 3   864   16
 0   0   0  -- 3   881   16
 0   0   0  -- 3   897   16
 0   0   0  -- 3   904   16
 0   0   0  ---1   0   0
```

Modulo 3.5.5: Riproduzione delle note

Questo modulo non è così arzigogolato come può sembrare a prima vista. È costituito in pratica da due copie dello stesso gruppo di linee, una per la voce uno e una per la voce due. La stessa cosa avrebbe potuto essere ottenuta con un solo ciclo, ma la pratica ci ha insegnato che questo metodo produce, in termini di ritardo fra le note o fra le due voci, suoni distinti in modo più naturale.
Ciò che il modulo fa' è passare da una voce all'altra, secondo la lunghezza del motivo per ciascuna voce, attivando la voce che deve produrre la nota successiva. Così, se la prima nota della voce uno ha una durata di 32, mentre le prime quattro note per la voce 2 sono tutte di lunghezza 8, il modulo produce la prima nota sulla voce uno e poi le quattro note sulla voce due, e solo dopo aver fatto questo torna a prendere la nota immediatamente successiva della voce uno. Questo processo continua finché una delle voci ha finito. A questo punto il record della lunghezza del motivo suonato su di essa viene posto artificialmente ad un valore molto alto in modo che il modulo suoni sempre l'altra voce. Quando i dati sono finiti per entrambe le voci, questo modulo restituisce l'esecuzione al modulo di controllo.
La musica vera e propria viene prodotta da un semplicissimo comando, costituito da VOL e SOUND, che lavora sulle informazioni memorizzate nel vettore PLAY%.

Modulo 3.5.5: Linee 13000-13200

```
13000 REM*******************************
13010 REM SUONA IL MOTIVO
13020 REM*******************************
13030 FOR I=0 TO 1 : TL(I)=0 : NN(I)=0 : NEXT
I
13040 DO UNTIL TL(0)=99999 AND TL(1)=99999
13050 DO WHILE TL(0)<=TL(1)
13060 IF PLAY%(NN(0),0,0)=-1 THEN TL(0)=99999
: EXIT
13070 VOL PLAY%(NN(0),0,0)
13080 SOUND
1,PLAY%(NN(0),0,1),PLAY%(NN(0),0,2)
13090 NN(0)=NN(0)+1
13100 TL(0)=TL(0) + PLAY%(NN(0),0,2)
13110 EXIT : LOOP
13120 DO WHILE TL(1)<=TL(0)
13130 IF PLAY%(NN(1),1,0)=-1 THEN TL(1)=99999
: EXIT
13140 VOL PLAY%(NN(1),1,0)
13150 SOUND
```

```
2,PLAY%(NN(1),1,1),PLAY%(NN(1),1,2)
13160 NN(1)=NN(1)+1
13170 TL(1)=TL(1) + PLAY%(NN(1),1,2)
13180 EXIT : LOOP
13190 LOOP
13200 RETURN
```

Commento

Linea 13030: Il vettore NN conserva la posizione raggiunta da ciascuna voce in termini di numero di note suonate, cioè la posizione nel vettore PLAY% della nota successiva per quella voce. TL conserva invece la durata del motivo suonato fino a quel momento su ciascuna voce.

Linee 13040 e 13190: È difficile che la durata della voce sia un valore alto quanto 99999, che può essere quindi usato per indicare che la voce ha finito. Quando entrambe le voci hanno terminato, il modulo ha svolto interamente la sua funzione.

Linee 13050–13110: Queste linee prendono e suonano una nota sulla voce uno, purché la durata del suo motivo (TL) sia minore o uguale a quella della voce due. Se si incontra un meno uno, TL(O) viene posto a 99999 e il ciclo ha termine. In caso contrario, PLAY% fornisce le informazioni per il comando VOL e SOUND. Non appena la nota è stata suonata, il valore di NN viene incrementato e la durata della nota sommata alla durata del motivo per la voce uno.

Linee 13120-13180: Esattamente uguali alle linee 13050-13110, con la sola differenza che qui abbiamo a che fare con la voce due. Come già detto, queste due serie di linee potrebbero essere combinate in un ciclo, ma, a nostro giudizio, tenendole separate si ottiene un certo incremento della velocità.

Verifica

Battete il comando RUN. Dopo una breve attesa, dovreste sentire due voci, una di un'ottava ascendente lenta, l'altra di due ottave ascendenti diverse suonate a velocità doppia. Per avere un'idea più chiara di quanto è possibile ottenere dal programma, provate ad introdurre un motivo un po' più lungo o inserite una linea provvisoria come, ad esempio:

```
11115 GOSUB 13000:GOTO 11115
```

che ripete la prova all'infinito.

Uso del programma

Come gli altri programmi di questo capitolo, anche questo può essere usato per trarre il meglio dalle prestazioni di cui è dotato il C16 nel campo della generazione del suono, o per rendere più vivi e brillanti gli altri programmi.
Se avete inserito la linea provvisoria proposta nell'ultima parte dell'ultimo modulo, avrete capito che, nonostante occorra un certo tempo per calcolare il motivo, una volta calcolato può essere suonato immediatamente.

La tecnica dell'uso della musica in altri programmi, quindi, non è tanto quella di inserire tutte le linee DATA e le linee che eseguono i calcoli, ma quella di memorizzare su disco il vettore PLAY% finito (il modulo di verifica alla linea 50000 può essere usato come modello per questo) e inserire nei programmi successivi solo le linee del modulo 3.5.5. Per suonare il motivo memorizzato, richiamate semplicemente il vettore PLAY% dal disco e poi usate il modulo 3.3.5. Tutto questo non richiede molta memoria, ma sta' a voi decidere se risparmiare o meno memoria, quando scrivete un programma. Non ci sono dubbi però che un po' di musica può dare a qualsiasi programma il tocco da maestro.

CAPITOLO 4
SEMPRE PIÙ SERIAMENTE

A questo stadio della vostra preparazione, dovreste ormai conoscere abbastanza bene le capacità del vostro C16 e le tecniche che possono consentirvi di metterle al lavoro per i vostri scopi. È arrivato dunque il momento di esaminare alcuni programmi sostanziali che permetteranno al vostro C16 di fare ciò per cui i microcomputer sono maestri, cioè gestire, classificare e richiamare le informazioni per conto dell'utente.
Di programmi, in questo capitolo, ne vedremo tre:

UNIFILE, un sistema di archiviazione molto capace, in grado di memorizzare una vasta gamma di informazioni facilmente riutilizzabili.

NNUMBER, un programma che crea un dizionario di nomi e numeri per qualsiasi cosa, permettendo all'utente di ottenere dalle fatture alle quotazioni di borsa fino, addirittura, al conteggio delle calorie per la propria dieta giornaliera.

TEXTED, un programma di word processing che lavora con il BASIC.

Programma 4.1: UNIFILE

Questo è davvero un gioiello di programma. È stato sviluppato negli anni, lavorando alla serie di libri "Working Micro". Coloro che hanno letto i libri che hanno preceduto questo, ci hanno scritto per dirci che lo usano nel lavoro, a scuola, per insegnare ai ragazzi come i micro gestiscono le informazioni, per organizzare i dati di associazioni ed enti o semplicemente per tenere in ordine libri o dischi.
L'attuale versione del C16 non ha niente di meno delle versioni precedenti, anche se, ovviamente, la quantità di informazioni che possono essere conservate nella memoria del C16 è molto inferiore rispetto a quelle che possono essere fatte stare su macchine più grandi come il Commodore Plus 4, il fratello maggiore del C16. Anche così, comunque, il programma attuale lascia più di 7000 byte di memoria a disposizione delle informazioni, e per la maggior parte degli usi che la gente fà di questo programma, questo significa la possibilità di memorizzare da cento a duecento voci per singolo file.
Esaminando questo programma, introdurremo due nuovi concetti:

1) La ricerca binaria
2) La compattazione di dati in stringhe continue

Figura 4.1: Tipico video nel modo ricerca dell'Unifile

```
voce  1 :-
cognome:rossi
nome:mario
indirizzo:corso garibaldi 36
telefono:4537635

>return Passa al data successivo
>'aaa' Per correggere
>'ccc' Per continuare la ricerca
>'#' col n.ro Per muovere il Puntatore
>'zzz' torna al menu

la tua richiesta:
```

Modulo 4.1.1: Impostazione della struttura del file

Molti libri destinati ai possessori di home computer contengono programmi di archiviazione di solito molto poco flessibili nelle loro modalità di impiego, in cui è insito che ogni volta che l'utente memorizza i dati personali di un certo nominativo, questi debbano essere collocati sotto le intestazioni Nome e Cognome, Indirizzo, Numero di telefono, o nell'ambito di una struttura analoga.
Il bello dell'Unifile è che, mentre consente certamente di usare una struttura di questo tipo, non impedisce di creare altri file in base a strutture diverse (da una a dieci intestazioni), senza che, per questo, il programma debba subire grosse modifiche. Unifile è ciò che potremmo definire un "camaleonte": cioè un programma in grado di adattarsi ad un gran numero di impieghi diversi, rispondendo all'utente in modi diversi in base al compito che stà eseguendo in quel momento.
Lo scopo di questo primo modulo è di inizializzare alcune variabili e di permettere il richiamo dei dati da disco, ma, cosa ancora più importante, la predisposizione del file originale nel modo desiderato.

Modulo 4.4.1: linee 10000-10130

```
10000 REM*********************************
10010 REM STRUTTURA DEL FILE
10020 REM*********************************
10030 DO UNTIL IN : IN=1
10040 R$=CHR$(13) : DIM ARRAY$(100)
10050 SCNCLR : CHAR ,15,1,"[RVS
```

```
CN][RED]SCHEDARIO" : PRINT
10060 INPUT "[CD][BLK]DEVI CARICARE DA DISCO
(S/N):";Q$
10070 IF Q$="S" THEN GOSUB 15090 : EXIT
10080 INPUT "[CD][CD][BLU]QUANTE VOCI IN OGNI
REGISTRAZIONE:";X
10090 DIM ITEM$(X-1),PTR(X-1)
10100 PRINT "[CD]"; : FOR I=0 TO X-1
10110 PRINT "[PUR]NOME DELLA
VOCE";STR$(I+1);":";:INPUT ITEM$(I)
10120 NEXT I
10130 LOOP
```

Commento

Linea 10040: Le informazioni destinate al programma Unifile vengono memorizzate nel vettore ARRAY$, che in questa linea, è dimensionato in modo da accettare fino a 101 accessi al file (o voci). Volendo però, il numero può essere aumentato a 200 o anche oltre. La cosa da ricordare e che ogni elemento del vettore, prima di essere usato, occupa tre byte di memoria. Se date al vettore le dimensioni di 500 voci, ricordate che usate fino a 1500 byte di memoria, più di un quinto della memoria disponibile, e questo prima ancora di memorizzarvi qualsiasi informazione. Sta a voi giudicare: se pensate di aver bisogno di meno di 100 accessi al file, potete lasciare l'istruzione DIM così com'è e risparmiare un po' di memoria preziosa.

Linee 10080-10120: Queste linee sono una parte del segreto della flessibilità dell'Unifile. Per ciascuna voce, che potete immaginare, se vi è di aiuto, come la singola scheda di uno schedario, potete definire da voi quanti titoli devono comparire. Se steste mettendo in ordine la vostra raccolta di dischi, potreste usare ad esempio i titoli: BRANO, TITOLO DELL'ALBUM, COMPOSITORE, ESECUTORE, DURATA. In questo caso, specifichereste cinque titoli e poi introdurreste i nomi per ciascuna. In futuro, ogni volta che doveste usare l'Unifile per memorizzare o richiamare informazioni nella vostra raccolta, vi verrebbe chiesto di introdurre un dato sotto ciascun titolo. Per quelli che non si lasciano spaventare dal linguaggio tecnico, quelle che abbiamo chiamato "voce" e "titolo", nel gergo dei computer, sono noti rispettivamente come "record" e "campo".

Modulo 4.1.2: Il menu

Un menu normale, con una particolarità, però: non esiste niente che faccia pensare alla fine del programma per mezzo di un TRAP o di un tasto STOP.

Unifile è un programma molto complesso e fermarlo con STOP a metà esecuzione potrebbe far sì che le informazioni che esso contiene vengano rovinate da un processo non completato. Per porre fine a Unifile è necessario ritornare sempre a questo menu e usare l'opzione quattro.

Modulo 4.1.2: Linee 11000-11140

```
11000 REM*******************************
11010 REM MENU
11020 REM*******************************
11030 Z=0 : DO UNTIL Z=4
11040 COLOR 0,8 : SCNCLR : CHAR ,16,1,"[RVS
ON][GRN]SCHEDARIO" : PRINT
11050 PRINT "[CD][CD][BLU]COMANDI A
DISPOSIZIONE:"
11060 PRINT,"[CD][CD][RED]1)IMMISSIONE DATI"
11070
PRINT,"[CD]2)RICERCA/VISUALIZZA/MODIFICA"
11080 PRINT,"[CD]3)REGISTRAZIONE DATI"
11090 PRINT,"[CD]4)STOP"
11100 INPUT"[CD][BLU]LA  TUA RICHIESTA:";Z
11110 ON Z GOSUB 12000,17000,18000
11120 LOOP
11130 SCNCLR : CHAR ,12,10,"[RVS
ON][BLK]SCHEDARIO CHIUSO"
11140 END
```

Modulo 4.1.3: File di dati

Non appena comincerete a sviluppare programmi più complessi, sia tratti da questo libro che ideati da voi, vi accorgerete di volere sempre più spesso introdurre il file dati il più presto possibile. Questo perché si possono eseguire verifiche attendibili solo introducendo quantità di dati abbastanza considerevoli. Poiché è molto probabile che commettiate degli errori e avrete bisogno di modificare delle linee, o vi rassegnate a reintrodurre sempre gli stessi dati a mano a mano che il programma viene sviluppato oppure scegliete la soluzione di memorizzarli sul disco fin dalle prime fasi di lavoro, richiamandoli da I0 per eseguire le prove necessarie.
Questo modulo vi permetterà di fare proprio questo, introducendo inoltre una possibilità di cui abbiamo già parlato prima, ma che non abbiamo ancora usato concretamente: dare al programma la capacità di prendere e memorizzare file con un gran numero di nomi.

Modulo 4.1.3: linee 18000-18160

```
18000 REM*******************************
18010 REM MEMORIZZAZIONE DATI
18020 REM*******************************
18030 INPUT "[CD]NOME DEL FILE";FI$
18040 FI$="@0:"+FI$+",S,W"
18050 OPEN 1,8,2,FI$ : PRINT#1,IT,R$,X
18060 FOR I=0 TO IT-1 : PRINT#1,ARRAY$(I) :
NEXT I
18070 FOR I=0 TO X-1 : PRINT#1,ITEM$(I) : NEXT
18080 PRINT#1 : CLOSE1 : RETURN
18090 INPUT "[CD]NOME DEL FILE";FI$
18100 FI$=FI$+",S,R"
18110 OPEN 1,8,0,FI$ : INPUT#1,IT,X : DIM
ITEM$(X-1),PTR(X-1)
18120 FOR I=0 TO IT-1
18130 GET#1,T$ : IF T$<>CHR$(13) THEN
ARRAY$(I)=ARRAY$(I)+T$ : GOTO 18130
18140 NEXT I
18150 FOR I=0 TO X-1 : INPUT#1,ITEM$(I) : NEXT
18160 CLOSE1 : RETURN
```

Commento

Linee 18030-18050 e 18090-18110: Il nome del file viene specificato in modo interattivo.

Linee 18050-18070: Il numero di voci gestite da Unifile in qualsiasi momento viene memorizzato nella variabile IT, mentre X, come probabilmente ricorderete, registra il numero di titoli (o campi) per ciascuna voce (o record). Queste linee registrano tutti gli elementi utili ricavati dal vettore ARRAY$ e i nomi dei titoli dal vettore ITEM$.

Linea 18130: Come potete notare, per acquisire i record memorizzati su disco, non abbiamo usato INPUT#, questo perché l'istruzione si limita alle stringhe con una lunghezza massima di 88 caratteri, e molte delle voci probabilmente sono destinate a superare tale limite. L'alternativa è di prendere un carattere per volta, usando l'istruzione GET#, fino al carattere RETURN che chiude la stringa.

Verifica

Impartite il comando RUN, rispondete N ai solleciti del drive e specificate quattro titoli scegliendo come nomi UNO, DUE, ecc.. Non appena ricompare il menu, scegliete l'opzione 3 e specificate un nome di file come, ad esempio, UNIDATI.

Il drive dovrebbe mettersi in funzione e subito dopo dovrebbe ricomparire il menu. Fermate il programma con l'opzione quattro e battete di nuovo il comando RUN. Questa volta rispondete Y al sollecito del drive e ribattete il nome di file dato prima. Non appena ricompare il menu, fermate di nuovo il programma e poi battete:

```
FOR I = 0 TO X-1:PRINT ITEM$(I):NEXT[RETURN]
```

e, come risultato, dovreste vedere sullo schermo i quattro campi scelti.

Modulo 4.1.4: Un metodo di ricerca migliore

In questo modulo, e nei due che seguono, daremo un'occhiata al modo in cui è possibile aggiungere una nuova voce al file principale contenuto nel vettore ARRAY$. La base del metodo, però, è contenuta in questo modulo che permette al programma Unifile di scorrere rapidamente fra le voci di un file fino a trovare la posizione dove inserire quella nuova o un ingresso da tastiera.
Il metodo è noto come "ricerca binaria" e può essere usato per ridurre sensibilmente il tempo di ricerca in qualsiasi programma che contenga lunghi elenchi di dati ordinati. Vediamo questo esempio:
Supponiamo che un file debba contenere 2000 nomi (poco probabile per un C16, ma non fa niente) e supponiamo di dovervi inserire un nuovo nome, rispettando l'ordine alfabetico. Guardando nella lista dei nomi, possiamo facilmente stabilire che il nuovo nome VERDI andrà inserito nella posizione 1731, anche se il computer, per ora, non ha alcuna possibilità di saperlo.
Una cosa che potremmo fare e imporre al computer di esaminare i nomi uno per uno dall'inizio. Comincerà dunque da ADAMI e deducerà che VERDI viene dopo, quindi passerà ad ADAMONI e così via. Dopo aver esaminato 1732 nomi, la ricerca raggiungerà il cognome VERDONI che viene dopo VERDI e sarà così stabilita la posizione corretta.
Questo è un metodo più che sicuro, ma non pensate che sarebbe meglio se il numero di confronti potesse essere ridotto un po'? Bene, nel caso del nostro file di 2000 nomi, l'intero processo può essere eseguito con appena 10 confronti. Come si fa? Semplice.
Il computer inizia la ricerca esaminando il nome della posizione 1024 del file, perché 1024 è la massima potenza di due (2^10) che può stare nel numero totale di nomi (2000). Il nome nella posizione 1024 risulta essere, dal punto di vista alfabetico, inferiore a VERDI e quindi il computer aggiunge 1024/2 o 512 o 2^9 alla posizione di ricerca iniziale, arrivando a 1536. Anche questa volta, il nome in questa posizione è inferiore a VERDI, perciò anche questa volta viene eseguita una somma, aggiungendo a 1536 il numero 256 o 2^8 e ottenendo il numero 1792. Adesso è successo qualcosa di nuovo, in quanto il nome nella posizione 1792 viene dopo VERDI in ordine alfabetico, perciò, invece della somma di 128, o 2^7, viene eseguita una sottrazione che dà per risultato il numero 1664.
La ricerca continua, attraverso somme e sottrazioni di potenze di due fino ad ottenere una struttura di questo tipo:

CONFRONTO N.	POSIZIONE	AZIONE
1	1024	+512
2	1536	+256
3	1792	-128
4	1644	+64
5	1728	+32
6	1760	-16
7	1744	-8
8	1736	-4
9	1732	-2
10	1730	+1

Provatelo per numeri diversi di voci e posizioni di inserimento e scoprirete che funziona sempre.

Modulo 4.1.4: Linee 13000-13100

```
13000 REM*******************************
13010 REM RICERCA BINARIA
13020 REM*******************************
13030 IF IT=0 THEN SS=0 : RETURN
13040 PO=INT(LOG(IT)/LOG(2)) : SS=2↑PO-1
13050 FOR I=PO-1 TO 0 STEP-1
13060
SS=SS+2↑I*((ARRAY$(SS)>TE$)-(ARRAY$(SS)<TE$))
13070 IF SS<0 THEN SS=0
13080 IF SS>IT-1 THEN SS=IT-1
13090 NEXT I
13100 IF ARRAY$(SS)<TE$ THEN SS=SS+1
13110 PO=0 : FOR I=1 TO LEN(TE$)
13120 IF MID$(TE$,I,1)="0" THEN PO=1
13130 NEXT I : SS=SS-PO
13140 RETURN
```

Commento

Linea 13020: Se nel file non ci fossero record, il calcolo provocherebbe l'insorgere di una condizione di errore. Questa linea evita che succeda.

Linea 13040: Queste due espressioni trovano la potenza maggiore di due che si adatta al numero di voci del file e poi stabiliscono che il puntatore di ricerca (SS) sia uguale a quel numero. Il "–1" nella seconda espressione tiene conto del fatto che il vettore inizia da zero e non da uno.

Linee 13050-13090: Questo ciclo esegue una ricerca, usando potenze decrescenti di due. Il file principale è contenuto nel vettore ARRAY$, la nuova voce da inserire, in TE$. Il valore del puntatore è fissato da due condizioni logiche che indicano se TE$ è maggiore o minore della voce nel vettore ARRAY$(SS).

Linee 13100-13140: Capita che la posizione cui si arriva sia inferiore o superiore di uno rispetto alla posizione giusta: in questo caso SS viene incrementato o decrementato di uno.

Modulo 4.1.5: Inserimento di un titolo (o campo)

Questo modulo ha il compito di inserire un nuovo campo nella posizione indicata dalla variabile SS. L'operazione viene eseguita semplicemente spostando tutto da SS in su' di una posizione.

Modulo 4.1.5: Linee 14000-14070

```
14000 REM*******************************
14010 REM INSERIMENTO
14020 REM*******************************
14030 IF IT=0 THEN GOTO 14070
14040 FOR I=IT TO SS+1 STEP -1
14050 ARRAY$(I)=ARRAY$(I-1)
14060 NEXT I
14070 ARRAY$(SS)=TE$ : IT=IT+1: RETURN
```

Modulo 4.1.6: Come introdurre nuovi record in un file

Dopo aver introdotto i moduli che eseguono l'effettivo lavoro, possiamo passare a quello con cui avrà contatti l'utente quando deve porre nuovo materiale nel sistema di archiviazione. La funzione di questo modulo è di chiedere all'utente di introdurre il giusto numero di campi, nel giusto ordine e per ciascun record, in modo da combinare i campi in un'unica stringa che si adatti ad una linea del vettore ARRAY$ e di chiamare poi i due moduli precedenti per inserire il record nel file principale.

Modulo 4.1.6: Linee 12000-12140

```
12000 REM*******************************
12010 REM REGISTRAZIONE NUOVA VOCE
12020 REM*******************************
12030 DO
12040 TE$="" : SCNCLR : CHAR ,12,1,"[RVS
ON][RED]IMMISIONE DATI" : PRINT
12050 PRINT,,"[CD][CR][GRN]";IT;" VOCI
```

```
IMMESSE"
12060 PRINT "[CD][BLU]COMANDI A DISPOSIZIONE:"
12070 PRINT "[CD][BLK] >[PUR]INSERISCI I DATI
NEI CAMPI"
12080 PRINT "[BLK] >[PUR]'ZZZ' RITONA AL
MENU[GRN][CD]"
12090 FOR I=0 TO X-1
12100 PRINT ITEM$(I)":"; : INPUT Q$
12110 IF Q$="ZZZ" THEN EXIT
12120 TE$=TE$+Q$+"[CD]" : NEXT I : PRINT
"[CD][BLK][FLASH ON]ATTENDI"
12130 GOSUB 13000 : GOSUB 14000
12140 LOOP
12150 RETURN
```

Commento

Linee 12090-12100: Queste sono le linee che richiedono l'introduzione del nuovo record. I nomi dei singoli campi vengono presi dal vettore ITEM$, che è stato predisposto nel modulo di inizializzazione. Tutti i campi vengono introdotti sotto il nome di Q$ e l'ingresso di quelli nuovi si interrompe non appena, in qualsiasi momento, Q$ si trasforma in ZZZ. Se l'input non è ZZZ, il campo viene aggiunto a TE$, che contiene l'intero record e alla fine del campo viene aggiunto un carattere di "cursore giù" ([CD]). Lo scopo di questo carattere non ha nulla a che fare con lo spostamento del cursore: esso viene usato semplicemente per indicare alle parti successive del programma dove inizia e finisce ogni campo. In questo modo, un record come, ad esempio, COGNOME, NOME, UNAVIA, UNACITTÀ, viene memorizzato nel vettore ARRAY$ in questa forma:

COGNOME[CD]NOME[CD]UNAVIA[CD]UNACITTÀ[CD]

costituendo così quella che viene chiamata "stringa compattata". Il motivo per cui è stato scelto il carattere [CD] è che è quasi impossibile che a qualcuno venga in mente di inserire un carattere di controllo del cursore in un accesso ad un file (se volete, comunque, potete scegliere un altro carattere di separazione).

Verifica

Siete ora in grado di verificare la correttezza degli ultimi tre moduli appena introdotti. Eseguite il programma e impostate il file con due campi per record, chiamati "UNO" e "DUE".

Quando avete finito l'inizializzazione del programma e siete fermi davanti al menu principale, scegliete l'opzione 1. Vi troverete davanti al video creato da questo modulo e vi verrà chiesto di introdurre il campo UNO. Battete AA1. Lo stesso sollecito sarà ripetuto per il campo DUE e batterete AA2. Ripetete la procedura

rispondendo ai successivi solleciti: DD1, DD2, CC1, BB1, BB2. Ora battete ZZZ e ritornerete al menu principale. Scegliete l'opzione quattro interrompendo così il programma e poi battete:

```
FOR I = 0 TO 3:? ARRAY$(I):NEXT[RETURN]
```

e dovreste veder comparire i dati seguenti:

```
AA1
    AA2
BB1
    BB2
CC1
    CC2
DD1
    DD2
```

Il motivo per cui il secondo elemento è visualizzato una linea più in basso e c'è una linea sullo schermo vuota è che stiamo riproducendo anche i caratteri [CD]. Normalmente essi verrebbero eliminati.
Se tutto ha funzionato perfettamente, potete far ripartire il programma con un'istruzione GOTO 11000 e, usando l'opzione tre del menu, memorizzare su disco i dati introdotti. Questo renderà le prossime prove meno lunghe e noiose.

Modulo 4.1.7: Identificazione dei campi di ogni singolo record

Prima di passare ai moduli che permettono il richiamo e la manipolazione dei dati richiamati da un file, abbiamo bisogno di questo modulo, la cui funzione consiste nel cercare un record e registrare la posizione di tutti i caratteri "CD" o, di fatto, la fine di ogni campo. I valori sono memorizzati nel vettore PTR, il quale contiene unicamente valori relativi a questo record. Quando viene richiesto un altro record, per poterlo analizzare sarà necessario che il modulo venga richiamato.

Modulo 4.1.7: Linee 19000-19080

```
19000 REM*******************************
19010 REM ANALISI DEL RECORD
19020 REM*******************************
19030 PP=0
19040 FOR I=0 TO X-1
19050 PTR(I)=INSTR(ARRAY$(S1),"[CD]",PP+1)
19060 PP=PTR(I)
19070 NEXT I
19080 RETURN
```

Commento

Linea 19050: Oltre al modulo di ricerca binaria, dove per indicare il record esaminato, viene usata la variabile SS, il resto del programma fa' uso della variabile S1 nella quale registra la posizione del record corrente nel vettore ARRAY$. La funzione della linea è di cercare tutti i caratteri [CD], cominciando un carattere dopo il punto in cui è stato trovato quello immediatamente precedente.

Modulo 4.1.8: Ricerca dei campi del file

Possiamo passare ora al modulo che rende il programma utilizzabile permettendo il richiamo di dati precedentemente memorizzati. Questo modulo permette di eseguire l'operazione in quattro modi possibili:

1) un campo dopo l'altro a partire dalla posizione attuale
2) saltando avanti e indietro per il numero specificato di campi
3) battendo un campo (il primo del record) per una ricerca più rapida
4) cercando tutte le volte in cui è presente una certa combinazione di caratteri, ovunque si trovi nell'ambito di un certo record.

Modulo 4.1.8: Linee 17000-17420

```
17000 REM*********************************
17010 REM RICERCA
17020 REM*********************************
17030 S1=0 : SCNCLR : CHAR ,16,1,"[RED][RVS
ON]RICERCA" : PRINT
17040 DO UNTIL IT>0
17050 PRINT"[CD]NESSUN DATO ANCORA IMMESSO" :
FOR I=1 TO 2000 : NEXT : EXIT
17060 LOOP : IF IT=0 THEN RETURN
17070 PRINT "[CD][BLU]COMANDI A DISPOSIZIONE:"
17080 PRINT "[CD][BLK]>[GRN]INSERIRE LE VOCI
PER LA RICERCA NORMALE"
17090 PRINT "[CD][BLK]>[GRN]PREMETTERE '*' PER
LA RICERCA INIZIALE"
17100 PRINT"[BLK]>[GRN][RVS ON]RETURN[RVS OFF]
PER IL PRIMO DATO DEL FILE"
17110 PRINT
"[CD][CD][PUR]"CHR$(113)CHR$(113)CHR$(113)CHR$
(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(
113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(1
13)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(11
3)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113
)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)
```

```
CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)C
HR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CH
R$(113);
17120 TE$="" : INPUT"[CD][CD][BLU]INSERISCI IL
COMANDO PER LA RICERCA:";TE$
17130 IF LEFT$(TE$,1)="*" THEN TE$=MID$(TE$,2)
: GOSUB 13000 : TE$="" : S1=SS
17140 P$="CCC" : DO WHILE P$="CCC"
17150 DO WHILE TE$<>""
17160 FF=0 : FOR I=S1 TO IT-1
17170 PP=INSTR(ARRAY$(I),TE$)
17180 IF PP<>0 THEN FF=1 : S1=I : I=IT-1
17190 NEXT I
17200 IF FF=0 THEN RETURN
17210 EXIT : LOOP
17220 P$="" : DO WHILE P$="" AND IT>0
17230 IF S1>IT-1 THEN S1=IT-1
17240 IF S1<0 THEN S1=0
17250 GOSUB 19000
17260 PRINT "[CLEAR][CD][RED]VOCE
";S1+1;":-[CD]":PP=0
17270 FOR I=0 TO X-1
17280 PRINT
"[GRN]";ITEM$(I);":[BLU]";MID$(ARRAY$(S1),PP+1
,PTR(I)-PP-1)
17290 PP=PTR(I) : NEXT I
17300 CHAR,1,15,"[RVS ON][RED]RICERCA[RVS OFF]
[BLU]COMANDI A DISPOSIZIONE:" : PRINT
17310 PRINT "[CD] [BLK]>[PUR][RVS
ON]RETURN[RVS OFF] PASSA AL DATA SUCCESSIVO"
17320 PRINT " [BLK]>[PUR]'AAA' PER CORREGGERE"
17330 PRINT " [BLK]>[PUR]'CCC' PER CONTINUARE
LA RICERCA"
17340 PRINT " [BLK]>[PUR]'#' COL N.RO PER
MUOVERE IL PUNTATORE"
17350 PRINT " [BLK]>[PUR]'ZZZ' TORNA AL MENU"
17360 INPUT "[CD][BLU]LA TUA RICHIESTA:";P$
17370 IF P$="" OR P$="CCC" THEN S1=S1+1
17380 IF P$="CCC" THEN EXIT
17390 IF P$="AAA" THEN GOSUB 15000 : P$=""
```

```
17400 IF LEFT$(P$,1)="#" THEN
S1=S1+VAL(MID$(P$,2)) : P$=""
17410 LOOP
17420 LOOP
17430 RETURN
```

Commento

Linea 17030: Come abbiamo già detto nel commento al modulo precedente, S1 è il puntatore al record attuale ed inizia da zero ogni volta che parte una ricerca.

Linee 17040-17060: Messaggio di errore generato nel caso in cui nel file non ci siano ancora delle voci.

Linea 17070-17120: Questo è il menu di partenza del modulo di ricerca. Ad ogni ricerca, lo vedrete comparire una volta sola, all'inizio. Usando questo menu, è possibile specificare il tipo di ricerca da eseguire (volendo modificare il tipo di ricerca, dovrete abbandonare quella appena avviata e iniziare di nuovo da questo menu). Il termine RICERCA NORMALE indica la ricerca di una data combinazione di caratteri (il primo record fornito sarà il primo del file che contiene quei caratteri), in qualsiasi posizione. RICERCA INIZIALE, invece, si riferisce alla ricerca di un record che inizia con la combinazione di caratteri specificati dall'utente. Così, l'uso del comando *SMI, porterebbe alla ricerca di un record che inizia con SMI, anche se non è il primo. Se la stringa specificata non è reperibile all'inizio di un qualsiasi record, al termine della ricerca viene fornito quello che occupa il posto nel quale verrebbe inserito se venisse introdotto come nuovo record.
Il programma vi dà la possibilità di ampliare sia la ricerca normale che quella iniziale su più di un campo del record: basta inserire un carattere [CD] nella stringa da trovare. Questa possibilità è conseguibile premendo SHIFT/INSERT e poi [CD]. In questo caso il risultato è la comparsa nella stringa da trovare di un carattere Q in negativo. Usando questa tecnica su un file in cui il primo campo di ciascun record è un cognome e il secondo è un nome, la ricerca iniziale di COGNOME[CD]A fornirebbe un COGNOME qualsiasi con un nome che comincia per A, che non necessariamente è il primo.
Nel caso della ricerca normale, se la stringa specificata risulta irreperibile in tutti i record del file, l'esecuzione del programma ritorna alla visualizzazione del menu del programma principale.

Linea 17130: Questa linea esegue tutto il lavoro necessario per una ricerca iniziale, togliendo l'asterisco e chiamando semplicemente il modulo di ricerca binaria per trovare la posizione corretta.

Linee 17140 e 17420: Il ciclo principale che ripete una ricerca normale nel caso in cui l'utente ne faccia richiesta in un menu successivo. Come potete notare, la routine della ricerca iniziale non rientra in questo ciclo: il fatto è che non c'è

ragione di ripetere una ricerca iniziale perché il risultato sarebbe dato sempre dallo stesso record.

Linee 17150-17210: A questo punto dell'esecuzione del modulo, qualsiasi input deve essere una stringa da cercare con il metodo di ricerca normale. L'operazione viene eseguita molto semplicemente, passando in rassegna tutti i record con l'istruzione INSTR. Non appena il modulo trova un campo, per indicarlo assegna alla variabile FF il valore uno. La posizione viene registrata in S1 e alla variabile I viene assegnato il suo valore più alto in modo che il ciclo termini.

Linee 17220 e 17410: Questo ciclo continua fintantoché P$, l'input nel menu di ricerca successivo, rimane su una stringa nulla. La ragione per cui il valore di IT deve essere incluso nella condizione per il ciclo è che, nell'ambito di quest'ultimo, sarà prevista la cancellazione dei campi. Se tutti i campi devono essere cancellati, avremo bisogno di uscire dal ciclo oppure il programma darà vita ad una condizione di errore.

Linee 17320-17240: Nell'ambito del ciclo, l'utente ha la possibilità di spostarsi nel file per numero (queste linee controllano che, nelle successive esecuzioni del ciclo, il puntatore non superi i limiti stabiliti per il numero attuale di record.

Linee 17250-17290: Dopo aver chiamato il modulo precedente, queste linee utilizzano le informazioni relative alla posizione dei caratteri di separazione per stampare i campi che costituiscono il record corrente, assieme al titolo del campo. Ad ogni esecuzione del ciclo FOR, vengono stampati i caratteri con una posizione fra il valore della variabile PP e un valore contenuto nel vettore PTR. Quando sono stati stampati tutti i gruppi di caratteri, a PP viene di nuovo assegnato il valore corrente del vettore PTR e la variabile I prende il valore successivo di quest'ultimo.

Linee 17300-17360: Il menu che compare una volta che un certo record è stato visualizzato. L'utente ha la possibilità di spostarsi sul record successivo, di chiamare la funzione MODIFICA (non ancora introdotta), di cercare la stringa specificata in precedenza, di spostare lungo il file un certo numero di record o, infine, di ritornare al menu del programma principale.

Linee 17370-17380: Queste due linee si riferiscono al ciclo che inizia dalle linee 17140 e 17220. Se l'input è una stringa nulla (cioè se viene premuto il tasto RETURN), il ciclo alla linea 17220 viene ripetuto stampando il record successivo indicato da S1. Se si batte invece CCC, il ciclo 17140 esegue di nuovo la ricerca a partire dal record immediatamente successivo.

Linea 17390: La funzione modifica, che non è stata ancora introdotta.

Linea 17400: Introducendo un numero preceduto dal segno "#", l'utente ha la possibilità di spostarsi avanti o indietro lungo il file. Assegnando a P$ una stringa nulla, si ottiene l'esecuzione del ciclo alla linea 17220 che genera il record cui il ciclo è arrivato.

Verifica

Se avete già salvato la serie di quattro record delle prove precedenti, eseguite il programma e ricaricate i record. Scegliete quindi l'opzione due del menu principale e, quando compare il menu RICERCA, fate scorrere con il tasto RETURN tutto il file. I record devono essere visualizzati su due linee con l'apposito nome di campo, ad esempio:

UNO: AA1
DUE: AA2

Appena raggiunto il quarto record, scoprirete che il tasto RETURN non vi permette più di continuare. Battete allora # –1 e dovreste potervi spostare indietro fino al record tre. Continuate a retrocedere finché scoprirete che oltre l'inizio del file è impossibile andare.
Tornate al menu principale battendo ZZZ e scegliete ancora l'opzione 2. Questa volta, rispondete al menu della ricerca iniziale battendo CC. Questo dovrebbe provocare la comparsa del record tre, il solo che contiene i caratteri CC. Vi trovate ora nel secondo menu di RICERCA, perciò continuate la ricerca battendo CCE e scoprirete che in questo modo ritornate al menu principale.
Ancora una volta scegliete l'opzione due, ma questa volta, come elemento da cercare, battete un 2. Vedrete comparire il record uno, dato che contiene il carattere 2. Per continuare la ricerca, battete CCC e vedrete comparire il record due (anch'esso contiene il carattere due). Continuate a battere CCC finché non leggerete sullo schermo tutti e quattro i record e la ricerca terminerà riconducendovi al menu principale.
Per finire, scegliete ancora una volta l'opzione due e battete *B, come elemento da trovare. Dovreste veder comparire il record due, cioè il solo record che inizi con B. A questo punto ritornate al menu principale, battendo ZZZ, e ponete fine al programma.
Avete così provato tutte le funzioni di ricerca.

Modulo 4.1.9: Cancellazione di record

Il tocco finale al programma è costituito da questi due moduli, che permettono la modifica o la cancellazione di record già esistenti. Vediamo per primo il modulo di cancellazione CANCELLA, dato che viene usato ogni volta che un record viene modificato.

Modulo 4.1.9: Linee 16000-16040

```
16000 REM*******************************
16010 REM CANCELLA LE VOCI
16020 REM*******************************
16030 FOR J=S1 TO IT-1 : ARRAY$(J)=ARRAY$(J+1)
: NEXT J
16040 IT=IT-1 : RETURN
```

Commento

Linee 16030-16040: La cancellazione viene eseguita semplicemente a cominciare dal record sopra quello da cancellare e copiandolo più in basso di una posizione. Quando tutti i record sono stati spostati, il loro numero si è ridotto di un'unità.

Verifica

Battete il comando RUN, ricaricate da disco i quattro record e poi interrompete il programma, scegliendo l'opzione quattro. Battete quindi:

S1 = 0[RETURN]
GOTO 16000

Il programma dovrebbe fermarsi generando il messaggio RETURN WITHOUT GOSUB. Battete allora:

GOTO 11000[RETURN]

e chiamate l'opzione RICERCA. A questo punto il record AA1/AA2 dovrebbe essere stato cancellato dal file.

Modulo 4.1.10: Modifica dei record

Il programma sarebbe di ben scarsa utilità se non permettesse di modificare i dati esistenti. Ebbene questo modulo è l'elemento del programma che ci consente di farlo.

Modulo 4.1.10: Linee 15000-15190

```
15000 REM*******************************
15010 REM MODIFICA REGISTRAZIONI
15020 REM*******************************
15030 TE$="" : PP=0
15040 FOR I=0 TO X-1
15050 PRINT "[CLEAR][CD][RED]VOCE ";S1+1;":-"
15060 PRINT
"[CD][GRN]";ITEM$(I);":[BLU]";MID$(ARRAY$(S1),
PP+1,PTR(I)-PP-1)
15070 CHAR ,16,13,"[RED][RVS ON]MODIFICA" :
PRINT
15080 PRINT "[CD][BLU]COMANDI DISPONIBILI:"
15090 PRINT "[CD] [BLK]>[GRN][RVS
ON]RETURN[RVS OFF] CONFERMA IL DATO ESISTENTE"
15100 PRINT " [BLK]>[GRN]INSERIRE IL NUOVO
DATO IN SOSTITUZIONE"
15105 PRINT "  A QUELLO VISUALIZZATO"
```

```
15110 PRINT " [BLK]>[GRN]'DDD'CANCELLA TUTTI I
DATI"
15120 PRINT " [BLK]>[GRN]'ZZZ'CONFERMA TUTTI I
DATI"
15130 Q$="" : INPUT "[CD][BLU]LA TUA
SCELTA:";Q$
15140 IF Q$="DDD" THEN GOSUB 16000 : RETURN
15150 IF Q$="ZZZ" THEN RETURN
15160 IF Q$<>"" THEN Q$=Q$+"[CD]"
15170 IF Q$="" THEN
Q$=MID$(ARRAY$(S1),PP+1,PTR(I)-PP)
15180 PP=PTR(I) : TE$=TE$+Q$ : NEXT I : GOSUB
16000 : GOSUB 13000
15190 S1=SS : GOSUB 14000 : RETURN
```

Commento

Linee 15040 e 15180: Questo ciclo, benché simile a quello che riproduce i record su schermo nel modulo 4.1.8, se ne differenzia nel senso che ne riproduce solo uno per volta.

Linea 15140: La battitura delle tre D, durante la presenza sul video di un campo, provoca la cancellazione del record di cui esso fa parte. Notate che non è possibile cancellare il singolo campo in quanto il numero di campi per record è fisso.

Linea 15150: Battendo ZZZ come valore da introdurre in un campo si ottiene come risultato il ritorno dell'esecuzione al modulo RICERCA. Qualsiasi cambiamento apportato ai campi precedenti di un record viene ignorato e il record rimane invariato.

Linea 15160: Se viene introdotto un dato diverso da DDD o ZZZ, questo viene interpretato come sostituzione del campo visibile. Alla fine del campo viene aggiunto il separatore [CD], come nel modulo 4.1.6.

Linea 15170: Premendo RETURN, senza input, si ottiene la copiatura del campo visualizzato senza alcun cambiamento. Se è necessario cambiare un solo campo, basta premere RETURN in risposta a tutti gli altri. Notate la differenza di uno fra questa espressione di stringa e quella usata per stampare il campo alla linea 15060 I – 1 alla fine viene eliminato in modo che non lo sia il carattere [CD].

Linee 15180-15190: Il record modificato viene ricomposto in TE$ e quando la modifica è stata eseguita, il record originale viene cancellato dal file. Il motivo per cui viene adottata questa procedura è che i cambiamenti apportati possono aver modificato la posizione corretta del record nel file ordinato secondo un certo criterio. Una volta cancellato, il record viene inviato al modulo di ricerca binaria

e reinserito, con la sua posizione nel file copiata nella variabile S1, in modo che il modulo RICERCA sappia quale campo visualizzare se la posizione è stata modificata.

Verifica

Eseguite il programma e ricaricate i quattro record dal disco. Se ora scegliete l'opzione di ricerca e premete il tasto RETURN, otterrete sullo schermo il record uno. In risposta al secondo menu RICERCA, battete tre volte A e vedrete comparire il primo campo AA1 assieme al menu MODIFICA. Battete AAA1 e, non appena compare il secondo campo, premete il tasto RETURN. Sarete così ritornati al modulo RICERCA e vedrete il record:

UNO:AAA1
DUE:AA2

Provate a fare qualche altro cambiamento e a cancellare altri record. Se tutto funziona correttamente, il programma può dirsi completo.

PROGRAMMA 4.2: NNUMBER

Non tutti i file hanno a che fare con le parole. Una delle cose che i microcomputer sanno fare molto bene è la memorizzazione e la manipolazione dei numeri. Questo secondo programma, il cui nome NNUMBER sta' per Nome e Numero, permette di memorizzare i nomi dei campi, le unità in cui sono di solito misurati e una quantità ad essi associata. Ora, prima che diciate che un programma di questo tipo non sarebbe per voi di alcuna utilità, provate ad immaginare di essere un negoziante o il cuoco di casa.
Il primo ha una quantità enorme di prodotti che costituiscono il suo magazzino. Tutti hanno un loro nome, vengono trattati in unità (scarole, bottiglie, sacchetti, ecc.) e hanno una quantità molto importante che si riferisce ad essi, vale a dire il prezzo. Per poter quindi usare un microcomputer in modo che possa aiutarlo in negozio, nella gestione delle scorte o per emettere fatture, bisogna che il nostro ipotetico negoziante tenga conto di tutti questi elementi del prodotto. Altro esempio. In casa, i cibi che mangiamo hanno un nome, vengono usati in diverse unità di peso (cucchiaio, bicchiere, pizzico, ecc..) e se ci interessa sapere il loro effetto sul nostro peso, è bene che a ciascuno sia associata una quantità di calorie.
Questi sono solo due esempi, ma potete pensare a tante situazioni diverse in cui poter registrare certi nomi con le relative unità e quantità può essere molto utile. Lo scopo del programma NNUMBER è quello di creare un dizionario di voci (fino a 200) assieme all'unità in cui sono misurate e al valore relativo a ciascuna.
Sulla base di questo dizionario, sarete in grado di costruire interi elenchi di voci che il programma visualizzerà assieme ai totali calcolati su richiesta.
NNUMBER è facile da usare tanto per calcolare il numero di calorie della propria dieta giornaliera quanto per ottenere il prezzo totale di un blocco di merci.

Figura 4.2: Esempio di visualizzazione nel modo ricerca del programma NNUMBER.

```
                    [illegible]

ARTICOLO NUMERO: 1

MAGAZZINO:PANTALONI

CONFEZIONE:SCATOLA

QUANTITA' PER SCATOLA: 10

●●●
```

```
(S/N):";Q$
10080 IF Q$="S" THEN GOSUB 21140 : EXIT
10090 INPUT "[CD][BLU]TIPO DI INVENTARIO:";NN$
10100 LOOP
```

Commento

Linea 10050: Il vettore ARRAY$ viene usato per registrare il nome dell'entità trattata e il nome dell'unità usata per ciascuna voce del dizionario. La quantità relativa ad ogni unità viene invece memorizzata nell'elemento equivalente del vettore ARRAY.
Il numero di voci che possono essere memorizzate nel dizionario è indicato dal valore della variabile MAX. Usare una variabile per assegnare le dimensioni ad un vettore può rappresentare una tecnica molto efficace, nel senso che, se si desidera in seguito modificarle, basta cambiare una sola cosa anche se per saperle occorrono più linee di programma (vedi linea 15050).
TE$ e TE vengono utilizzati con lo stesso scopo dei vettori ARRAY$ e ARRAY, ma per la lista "corrente" ricavata dal dizionario.

Linea 10060: CURR registra il numero di voci della lista corrente, cioè quella ricavata dal dizionario principale. IT, come nella maggior parte dei programmi di questo libro, registra il numero di voci del file principale (in questo caso, il dizionario).

Modulo 4.2.2: Menu

Un normale modulo menu. La sola differenza, rispetto ai programmi prescelti, è il modo in cui la linea 11150 impedisce che certe funzioni del programma siano usate prima dell'introduzione dei dati.

Modulo 4.4.2.: Linee 11000-11220

```
11000 REM*******************************
11010 REM MENU
11020 REM*******************************
11030 Z=0 : DO UNTIL Z=8
11040 SCNCLR : CHAR ,13,1,"[RED][RVS
ON]INVENTARIOE":PRINT
11050 PRINT "[CD][CD][BLU]COMANDI A
DISPOSIZIONE:"
11060 PRINT "[GRN][CD][CD]   1) VISUALIZZA
L'INVENTARIO"
11070 PRINT "   2) INSERIMENTO QUANTITA'"
11080 PRINT "   3) AZZERAMENTO TOTALE
QUANTITA'
11090 PRINT "   4) ELIMINA SINGOLA QUANTITA'"
```

```
11100 PRINT "    5) INSERIMENTO NUOVI ARTICOLI"
11110 PRINT "    6) MODIFICA ARTICOLI"
11120 PRINT "    7) REGISTRAZIONE DATI"
11130 PRINT "    8) STOP"
11140 INPUT "[CD][BLU]LA TUA SCELTA:";Z :
SCNCLR
11150 DO WHILE IT=0 AND (Z=1 OR Z=4 OR Z=6 OR
Z=7)
11160 PRINT "[CD][FLASH ON]NESSUN DATO ANCORA
INSERITO" : ND=1
11170 GETKEY A$ : EXIT : LOOP
11180 IF ND=0 THEN ON Z GOSUB
12000,13000,14000,20000,15000,18000,21000
11185 ND=0
11190 LOOP
11200 SCNCLR
11210 CHAR ,12,12,"[RED]NOME E NUMERO" : PRINT
11220 PRINT "[CD][CD]","[CL][CL][GRN]PROGRAMMA
  TERMINATO" : END
```

Modulo 4.2.3: Memorizzazione di dati

Anche questo è un modulo standard.

Modulo 4.2.3: Linee 21000-21240

```
21000 REM*********************************
21010 REM FILE DATI
21020 REM*********************************
21030 INPUT "[CD]NOME DEL FILE DA
REGISTRARE";FI$
21040 FI$="@0:"+FI$+",S,W"
21050 OPEN 1,8,2,FI$
21060 PRINT#1,NN$;R$;CU;R$;IT
21070 DO WHILE CU>0
21080 FOR I=0 TO CU-1 :
PRINT#1,TE$(I,0);R$;TE$(I,1);R$;TE(I) : NEXT
21090 EXIT : LOOP
21100 DO WHILE IT>0
21110 FOR I=0 TO IT-1 :
PRINT#1,ARRAY$(I,0);R$;ARRAY$(I,1);R$;ARRAY(I)
```

```
 : NEXT
21120 EXIT : LOOP
21130 PRINT#1 : CLOSE1 : RETURN
21140 INPUT "[CD]NOME DEL FILE DA
CARICARE";FI$
21150 FI$=FI$+",S,R"
21160 OPEN 1,8,2,FI$
21170 INPUT#1,NN$,CU,IT
21180 DO WHILE CU>0
21190 FOR I=0 TO CU-1 :
INPUT#1,TE$(I,0),TE$(I,1),TE(I) : NEXT
21200 EXIT : LOOP
21210 DO WHILE IT>0
21220 FOR I=0 TO IT-1 :
INPUT#1,ARRAY$(I,0),ARRAY$(I,1),ARRAY(I) :
NEXT
21230 EXIT : LOOP
21240 CLOSE1 : RETURN
```

Modulo 4.2.4: Ricerca binaria

Per avere il commento di questo modulo, rileggete quello relativo al modulo equivalente del programma UNIFILE. La sola cosa da notare è che l'ordinamento viene eseguito sulla base del nome nella colonna zero del vettore ARRAY$.

Modulo 4.2.4: Linee 16000-16110

```
16000 REM*******************************
16010 REM RICERCA BINARIA
16020 REM*******************************
16030 IF IT=0 THEN SS=0 : RETURN
16040 PO=INT(LOG(IT)/LOG(2)) : SS=2↑PO-1
16050 FOR I=PO TO 0 STEP-1
16060
SS=SS+2↑I*((ARRAY$(SS,0)>T1$)-(ARRAY$(SS,0)<T1
$))
16070 IF SS<0 THEN SS=0
16080 IF SS>IT-1 THEN SS=IT-1
16090 NEXT I
16100 IF ARRAY$(SS,0)<T1$ THEN SS=SS+1
16110 RETURN
```

Modulo 4.2.5: Inserimento di voci nel dizionario principale

Il principio su cui si basa questo modulo è identico a quello del modulo equivalente del programma UNIFILE. La ragione per cui questo è leggermente più lungo è che nei due vettori deve inserire due stringhe e un numero invece di una sola stringa.

Modulo 4.2.5.: Linee 17000-17120

```
17000 REM*******************************
17010 REM INSERIMENTO NUOVO ARTICOLO
17020 REM*******************************
17030 DO WHILE IT<>0
17040 FOR I=IT TO SS+1 STEP-1
17050 ARRAY$(I,0)=ARRAY$(I-1,0)
17060 ARRAY$(I,1)=ARRAY$(I-1,1)
17070 AR(I)=AR(I-1)
17080 NEXT : EXIT : LOOP
17090 ARRAY$(SS,0)=T1$
17100 ARRAY$(SS,1)=T2$
17110 AR(SS)=NN
17120 RETURN
```

Modulo 4.2.6: Introduzione di voci per il dizionario

Molto meno complicato del modulo equivalente del programma UNIFILE, questo modulo accetta tre dati dall'utente: a) il nome della voce, b) il nome delle unità in cui viene misurata, c) la quantità relativa ad ogni singola unità.

Modulo 4.2.6: Linee 15000-15140

```
15000 REM*******************************
15010 REM INSERIMENTO ARTICOLI
15020 REM*******************************
15030 DO
15040 PRINT"[CLEAR][CD]","[RVS ON]NUOVI
ARTICOLI"
15050 IF IT>MAX THEN PRINT"[CD][RED]NON C'E'
PIU' SPAZIO IN MEMORIA" : GETKEYA$ : EXIT
15060 Q$="" : DO UNTIL Q$="S"
15070 PRINT "[CD][CD][RED]";NN$:INPUT "(NOME O
'ZZZ' PER TORNARE AL MENU):";T1$
15080 IF T1$="ZZZ" THEN EXIT
15090 PRINT"[CD]";QQ$;:INPUT ":";T2$
15100 PRINT"[CD]QUANTITA' PER ";T2$;:INPUT NN
```

```
15110 INPUT "[CD][BLU]I DATI SONO ESATTI
(S/N):";Q$:IF Q$="N" THEN GOTO 15060
15120 LOOP : IF T1$="ZZZ" THEN EXIT
15130 GOSUB 16000 : GOSUB 17000 : IT=IT+1
15140 LOOP : RETURN
```

Verifica

E finalmente possiamo fare la prima vera prova di quanto è stato introdotto. Battete il comando RUN, comunicate al computer che non state per caricare da disco e fornite il nome ITEM in risposta alla richiesta di un nome generico. Scegliete nel menu principale l'opzione cinque (Inserimento nuovi articoli) e, quando compare sullo schermo la scritta "nuove voci", battete quanto segue:

COSA1/SCATOLA/10
COSA2/BOTTIGLIA/20
COSA3/SACCHETTO/40

Tutte queste voci non hanno nessun significato particolare: servono solo per eseguire la prova.
Terminata la battitura, ritornate al menu principale battendo ZZZ. Poi, per memorizzare le informazioni, chiamate il modulo registrazione dati (opzione sette), infine fermate il programma con l'opzione otto e battete:

FOR I = 0 TO 2:?ARRAY$(I,0),ARRAY$(I,1),ARRAY%(I):NEXT[RETURN]

e dovreste veder comparire quanto segue:

COSA1	SCAROLA	10
COSA2	BOTTIGLIA	20
COSA3	SACCHETTO	40

Ora eseguite il programma e comunicate che questa volta intendete caricare da disco. Date il nome che avete scelto al momento della memorizzazione dei dati e, quando il disco ha terminato, dovreste essere in grado di eseguire qualche prova sul contenuto dei vettori, con gli stessi risultati.

Modulo 4.2.7: La routine di ricerca

Come nel programma UNIFILE, questo modulo offre all'utente la possibilità di muoversi lungo il dizionario, alla ricerca di voci specifiche che possono, volendo, essere cancellate dal file. Nel suo complesso, è molto più semplice di quello equivalente proposto per il programma UNIFILE, dato che è stato studiato per cercare solamente voci complete, e non combinazioni di caratteri memorizzati qua e là in un elemento da trovare. Inoltre, la struttura di un record completo in questo programma è molto più semplice della struttura della voce nel vettore principale del programma UNIFILE.

Modulo 4.2.7: Linee 18000-18260

```
18000 REM*******************************
18010 REM RICERCA
18020 REM*******************************
18030 SS=0
18040 T1$="" : DO UNTIL T1$="ZZZ"
18050 DO
18060
PRINT"[CLEAR][CD][CR][CR][CR][CR][CR][CR][CR][
CR][CR][CR][CR][CR][CR][CR][RVS ON]RICERCAr"
18070 PRINT"[BLU][CD]ARTICOLO NUMERO:";SS+1
18080 PRINT"[CD][RED]";NN$;":";ARRAY$(SS,0)
18090 PRINT"[CD]";QQ$;":";ARRAY$(SS,1)
18100 PRINT"[CD]QUANTITA' PER
";ARRAY$(SS,1);":";AR(SS)
18110
PRINT"[CD][GRN]"CHR$(113)CHR$(113)CHR$(113)CHR
$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$
(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(
113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(1
13)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(11
3)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113
)CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)
CHR$(113)CHR$(113)CHR$(113)CHR$(113)CHR$(113)C
HR$(113)
18120 PRINT"[CD]COMANDI A DISPOSIZIONE:"
18130 PRINT"[CD][GRN]   >ARTICOLO DA
RICERCARE"
18140 PRINT"   >'#'NUMERO PER CAMBIARE
ARTICOLO"
18150 PRINT "   >[RVS ON]RETURN[RVS OFF] PER
L'ARTICOLO SEGUENTE"
18160 PRINT"   >'DDD' CANCELLA L'ARTICOLO"
18170 PRINT"   >'ZZZ' TORNA AL MENU"
18180 T1$="#1":INPUT "[CD][PUR]LA TUA
SCELTA:";T1$
18190 IF T1$="DDD" THEN GOSUB 19000 : EXIT
18200 IF T1$="ZZZ" THEN EXIT
18210 IF LEFT$(T1$,1)<>"#" THEN GOSUB 16000
```

```
18220 IF LEFT$(T1$,1)="#" THEN
SS=SS+VAL(MID$(T1$,2))
18230 IF SS>IT-1 THEN SS=IT-1
18240 IF SS<0 THEN SS=0
18250 LOOP
18260 LOOP : RETURN
```

Commento

Linee 18040, 18050, 18250 e 18260: La presenza di questi due cicli può sembrare a prima vista strana, ma se esaminate le linee 18190-18200, vedrete che offrono un metodo molto semplice per eseguire un azione nel ciclo più interno e poi saltare tutte le azioni successive di quel ciclo usando il comando EXIT. La linea 18190, che gestisce il comando di cancellazione, trasferisce il controllo dell'esecuzione fuori dal ciclo interno non appena la cancellazione è stata fatta.

Linea 18180: Invece dell'inserimento di un RETURN (cioè di una stringa vuota), viene prima assegnato il valore #1 alla variabile TI$. Se l'utente preme semplicemente RETURN, il contenuto di TI$ viene modificato e viene visualizzata la voce del file immediatamente successiva.

Verifica

Battete il comando RUN, ricaricate i tre dati dal disco e poi scegliete l'opzione sei (Modifica Articoli) del menu principale e, a questo punto, dovreste poter far scorrere l'elenco di voci, sia in avanti che indietro, usando un numero preceduto dal simbolo "#", come nel programma UNIFILE. Dovreste essere inoltre in grado di recuperare singole voci introducendone semplicemente il nome (non il nome dell'unità).

Modulo 4.2.8: Cancellazione di una voce

Equivalente diretto del modulo di cancellazione del programma UNIFILE.

Modulo 4.2.8: Linee 19000-19070

```
19000 REM*******************************
19010 REM CANCELLAZIONE
19020 REM*******************************
19030 FOR I=SS TO IT-1
19040 ARRAY$(I,0)=ARRAY$(I+1,0)
19050 ARRAY$(I,1)=ARRAY$(I+1,1)
19060 AR(I)=AR(I+1)
19070 NEXT : IT=IT-1 : RETURN
```

Verifica

Battete il comando RUN, ricaricate i dati, scegliete l'opzione sei del menu

principale e battete DDD in corrispondenza di uno dei record. Dovreste scoprire che è stato tolto dal file.

Modulo 4.2.9: Copiatura di voci nell'elenco corrente

Lo scopo del programma NNUMBER è non solo quello di tenere un dizionario di voci con le relative quantità, ma anche di poterlo usare come base su cui costruire liste provvisorie. Il modulo che segue è quindi studiato per permettere all'utente di aggiungere nuove voci nella lista corrente, di cancellare singole voci o la loro lista completa in un'unica operazione e infine di scegliere voci dal dizionario principale e copiarle nella lista usata al momento.

Modulo 4.2.9: Linee 13000-13170

```
13000 REM*******************************
13010 REM INSERIMENTO QUANTITA'
13020 REM*******************************
13030 DO : SCNCLR
13040 CHAR ,8,2,"[RED][RVS ON]INSERIMENTO
QUANTITA'" : PRINT
13050 IF CU>49 THEN PRINT,"[RED][CD]INVENRARIO
AL COMPLETO." : PRINT : GETKEY A$ : EXIT
13060 PRINT"[CD][CD][GRN]";NN$;" ('ZZZ' TORNA
AL MENU):"; : INPUT T1$
13070 IF T1$="ZZZ" THEN EXIT
13080 KN=0 : GOSUB 16000 : IF ARRAY$(SS,0)=T1$
THEN KN=1
13090 IF KN=0 THEN
PRINT"[CD][CD][BLU]"NN$"-ARTICOLO NON PRESENTE
-" : GETKEY A$ : EXIT
13100 PRINT"[CD]";QQ$;":";ARRAY$(SS,1) : INPUT
"[CD]QUANTITA':";Q
13110 INPUT "[CD][CD]I DATI SONO ESATTI
(S/N):";Q$
13120 DO WHILE Q$="S"
13130 TE$(CU,0)=ARRAY$(SS,0)
13140 TE$(CU,1)=STR$(Q)+" "+ARRAY$(SS,1)
13150 TE(CU)=Q*AR(SS) : CU=CU+1
13160 EXIT : LOOP
13170 LOOP : RETURN
```

Commento

Linee 13080-13090: Queste linee eseguono un controllo per vedere se la voce,

battuta dall'utente e destinata ad essere inserita nella lista in uso, esiste nel dizionario principale. Il controllo viene effettuato cercando, con il modulo di ricerca binaria, la posizione in cui la voce dovrebbe essere inserita nel dizionario. Il contenuto effettivo del dizionario a questo punto viene confrontato con la voce battuta dall'utente. Se quest'ultima è contenuta nel dizionario, significa che le due voci sono uguali, in caso contrario viene generato un messaggio di errore e il modulo ha termine.

Linea 13100: Dopo aver trovato la voce nel dizionario, il modulo riproduce su schermo le unità in cui essa viene normalmente misurata e chiede quante di queste unità devono essere inserite.

Linee 13110-13160: All'utente viene chiesto di confermare la precisione del record, prima che venga inserito nella lista in uso, contenuta nelle variabili TE$, e TE. Notate che la quantità memorizzata in TE non è la quantità per unità di misura ricavata dal dizionario, ma la quantità complessiva per il numero di unità specificate dall'utente.

Verifica

Eseguite il programma, ricaricate i dati da disco, scegliete nel menu principale l'opzione due e infine battete quanto segue:

```
COSA1,1
COSA2,2
COSA3,3
```

Ora provate a far accettare al modulo la voce COSA4, che non è inclusa nel dizionario. Come vi accorgerete immediatamente, il tentativo fallisce e infatti vedrete comparire un messaggio di errore e la richiesta a controllare il nome appena fornito. Premete un tasto qualsiasi e ritornerete al menu principale. Prima di fare qualsiasi altra cosa, scegliete l'opzione sette, con cui potrete memorizzare la linea appena creata assieme al dizionario principale. Per finire scegliete l'opzione otto e interrompete il programma.
Ora battete, nel modo diretto, questa linea:

```
FOR I = 0 TO2:?TE$(I,0),TE0$(I,1),TE(I):NEXT[RETURN]
```

e dovreste veder comparire questi dati:

```
COSA1    1    SCATOLA     10
COSA2    2    BOTTIGLIA   40
COSA3    3    SACCHETTO  120
```

Modulo 4.2.10: Visualizzazione della lista in uso

Il solo scopo di questo modulo è di riprodurre sullo schermo, uno per uno, i record che costituiscono la lista corrente. Dopo ogni voce introdotta, l'utente viene

sollecitato a premere un tasto, prima che sia visualizzata quella immediatamente successiva. Questo perché la lista è normalmente più lunga dello schermo e l'utente non potrebbe leggere un testo che scorresse più veloce di quanto consente la sua capacità di lettura. Alla fine dell'elenco il modulo fornisce il totale delle quantità relative al contenuto della lista in uso.

Modulo 4.2.10: Linee 12000-12150

```
12000 REM*******************************
12010 REM VISUALIZZA INVENTARIO
12020 REM*******************************
12030 DO WHILE CU>0
12040 SCNCLR : CT=0
12050 FOR I=0 TO CU-1
12060 PRINT"[BLU]";NN$;":";TE$(I,0)
12070 PRINTQQ$;":";TE$(I,1)
12080 PRINT"QUANTITA':";TE(I)
12090
PRINT"[GRN]"CHR$(119)CHR$(119)CHR$(119)CHR$(11
9)CHR$(119)CHR$(119)CHR$(119)CHR$(119)CHR$(119
)CHR$(119)CHR$(119)CHR$(119)CHR$(119)CHR$(119)
CHR$(119)CHR$(119)CHR$(119)CHR$(119)CHR$(119)C
HR$(119)CHR$(119)CHR$(119)CHR$(119)CHR$(119)CH
R$(119)CHR$(119)CHR$(119)CHR$(119)CHR$(119)CHR
$(119)CHR$(119)CHR$(119)CHR$(119)CHR$(119)CHR$
(119)CHR$(119)CHR$(119)CHR$(119)CHR$(119)
12100 GETKEY A$
12110 CT=CT+TE(I)
12120 NEXT I
12130 PRINT"[PUR][CD]TOTALE
QUANTITA':[BLK]";CT
12140 PRINT"[CD][GRN]PREMI UN TASTO PER
TORNARE AL MENU" : GETKEY A$
12150 EXIT : LOOP : RETURN
```

Modulo 4.2.11: Cancellazione di voci dalla lista in uso

Questa è una versione semplificata del tipo di modulo di ricerca usato per il dizionario principale. Esso consente all'utente di scorrere lungo l'elenco in uso, una voce per volta, e di cancellare qualsiasi voce battendo in corrispondenza DDD. L'elaborazione può essere terminata in qualsiasi momento con ZZZ.

Modulo 4.2.11: Linee 20000-20160

```
20000 REM*******************************
20010 REM CANCELLAZIONE ARTICOLO
20020 REM*******************************
20030 I=0 : DO WHILE I<CU : Q$=""
20040 PRINT"[BLU][CD]";TE$(I,0)
20050 PRINT"[CD]";TE$(I,1)
20060 INPUT "[GRN][CD]DDD=CANCELLA[RVS ON]
[RVS OFF]RETURN=PROSSIMO[RVS ON] [RVS
OFF]ZZZ=MENU:";Q$:IFQ$="ZZZ" THEN RETURN
20070 IFQ$="ZZZ" THEN EXIT
20080 IF Q$<>"DDD" THEN I=I+1
20090 DO WHILE Q$="DDD"
20100 FOR J=I TO CU-1
20110 TE$(J,0)=TE$(J+1,0)
20120 TE$(J,1)=TE$(J+1,1)
20130 TE(J)=TE(J+1)
20140 NEXT :CU=CU-1
20150 EXIT : LOOP
20160 LOOP : RETURN
```

Commento

Linea 20080: La variabile che controlla quale voce viene visualizzata è la variabile I. La ragione per cui non viene incrementata quando viene battuto DDD è che non è necessario (staremo già puntando alla voce da cancellare e il processo di cancellazione copierà la voce successiva in quella posizione).

Verifica

Eseguite il programma e ricaricate il file di dati contenente la lista corrente, che avete salvato durante la verifica del modulo 4.2.9. Scegliete quindi l'opzione quattro (Elimina Singola Quantità) del menu principale e dovreste essere in grado di scorrere lungo le tre voci dell'elenco corrente e di cancellarne una (il fatto che sia stata effettivamente cancellata può essere verificato visualizzando la lista in uso).

Modulo 4.2.12: Inizializzazione della lista in uso

Sono molte le applicazioni che richiedono la ricostruzione della lista attuale, il totale relativo e poi lo spostamento rapido su un elenco nuovo. Questo modulo molto semplice cancella il contenuto della lista corrente in una sola operazione.

Modulo 4.2.12: Linee 14000-14060

```
14000 REM*********************************
14010 REM ARREZAMENTO QUANTITA'UNITARIA
14020 REM*********************************
14030 FOR I=0 TO CU-1
14040 TE$(I,0)="" : TE$(I,1)="" : TE(I)=0
14050 NEXT I
```

Verifica

Eseguite il programma, ricaricate il file di dati che contiene la lista corrente e scegliete l'opzione tre (Azzeramento Totale Quantità) del menu principale. In un attimo, quest'ultimo dovrebbe ricomparire sullo schermo. Ora provate a scegliere l'opzione uno. Ancora una volta, tutto ciò che succede è la ricomparsa immediata del menu principale (il modulo di visualizzazione è stato chiamato ma, poiché non c'è nulla da visualizzare, l'esecuzione ritorna al punto di partenza). Se il test si è concluso senza errori, il programma può ritenersi completo e pronto per essere usato.

PROGRAMMA 4.3: TEXTED

Il fratello maggiore del C16, il PLUS FOUR, con una memoria disponibile cinque volte più grande, viene fornito con un chip incorporato che fornisce all'utente uno dei prodotti più affascinanti, vale a dire il word processor. Usando un word processor moderno e sofisticato, è possibile eliminare per sempre tutti i problemi connessi con la scrittura di testi, di qualsiasi natura essi siano, e anche la redazione dei documenti più complessi si trasforma in un compito quanto mai semplice. Purtroppo, però, un programma che sia in grado di eseguire tutte le funzioni di un word processor occuperebbe molta più memoria di quella disponibile con i 16K del C16.

Il programma che stiamo per proporvi, che abbiamo chiamato TEXTED, anche se non può essere definito un vero "word processor", permette di trattare il C16 come una macchina da scrivere un po' più sofisticata, in grado di comporre un testo, modificarlo, cancellare linee o parole, spostare linee qua e là e stampare il tutto su una stampante compatibile. Se state cercando qualcosa da usare in ufficio, beh, questo non fa' proprio al caso vostro, se invece volete qualcosa da usare per scrivere brevi lettere o semplici documenti, allora questo programma può essere lo strumento adatto (comunque una cosa che ci fa molto piacere è ricevere lettere di nostri lettori, i quali ci dicono con un certo orgoglio di aver scritto la loro lettera usando le versioni precedenti di questo stesso programma). Eccovi, per incominciare, gli argomenti che trattiamo per la prima volta:

1) Vantaggi dell'istruzione GET nell'introduzione di un testo
2) Cursori controllati da programma e l'input del testo
3) Manipolazioni di materiale in grandi vettori di stringa
4) Uscita su stampante

Modulo 4.3.1: Inizializzazione

Un modulo molto semplice e lineare, senza niente di particolare da notare salvo l'uso di una o due variabili dichiarate.

Modulo 4.3.1: Linee 11000-11130

```
11000 REM*******************************
11010 REM INIZIALIZZAZIONE
11020 REM*******************************
11030 IN=1
11040 DIM TEXT$(100) : LL=1 : PL=1
11050
TEXT$(0)="[BRN]"CHR$(127)CHR$(169)CHR$(127)CHR
$(169)CHR$(127)CHR$(169)CHR$(127)CHR$(169)CHR$
(127)CHR$(169)CHR$(127)CHR$(169)CHR$(127)CHR$(
169)CHR$(127)CHR$(169)CHR$(127)CHR$(169)CHR$(1
27)CHR$(169)CHR$(127)CHR$(169)CHR$(127)CHR$(16
9)CHR$(127)CHR$(169)CHR$(127)CHR$(169)CHR$(127
)CHR$(169)CHR$(127)CHR$(169)CHR$(127)CHR$(169)
CHR$(127)CHR$(169)CHR$(127)CHR$(169)CHR$(127)C
HR$(169)
11060 TEXT$(1)="[BRN][RVS
ON]"CHR$(169)CHR$(127)CHR$(169)CHR$(127)CHR$(1
69)CHR$(127)CHR$(169)CHR$(127)CHR$(169)CHR$(12
7)CHR$(169)CHR$(127)CHR$(169)CHR$(127)CHR$(169
)CHR$(127)CHR$(169)CHR$(127)CHR$(169)CHR$(127)
CHR$(169)CHR$(127)CHR$(169)CHR$(127)CHR$(169)C
HR$(127)CHR$(169)CHR$(127)CHR$(169)CHR$(127)CH
R$(169)CHR$(127)CHR$(169)CHR$(127)CHR$(169)CHR
$(127)CHR$(169)CHR$(127)CHR$(169)CHR$(127)"[RV
S OFF]"
11070 A$=" "
11080 FILL$="
        ":REM 40 SPAZI
11090 DEF FNA(P)=3072+20*40+P
11100 DEF FNB(P2)=3072+40*P2
11110 INPUT "DEVI CARICARE DA DISCO (S/N):";Q$
11120 IF Q$="S" THEN GOSUB 18140
11130 RETURN
```

Figura 4.3: Esempio di lettera composta e stampata usando il programma TEXTED

Egregio Signor
Mario Rossi
via Verdi, 3
20100 Milano

Egregio Signor Rossi,
Questo è un esempio di lettera composta usando il programma TEXTED.
Anche se come vedrà, le possibilità offerte sono molto elementari, i risultati che si possono ottenere sono davvero sorprendenti.
Lo provi e ne resterà soddisfatto.
Distinti saluti.

Commento

Linea 11040: Il vettore TEXT$ verrà usato per contenere il cosiddetto "testo principale".

Linee 11050-11060: Queste due linee di simboli grafici (che si trovano sui tasti della lira sterlina e dell'asterisco) formano due indicatori che compaiono sullo schermo non appena vengono toccati l'inizio o il fondo del blocco di testo.

Linea 11090: Se avete introdotto i programmi grafici de terzo capitolo, questa funzione dovrebbe esservi già nota. Il suo scopo è quello di indicare una posizione sullo schermo in relazione all'inizio della linea 20.

Linea 11100: Una funzione che ha a che fare con una posizione sullo schermo, solo che questa volta il valore fornito indica il primo quadro-carattere di una singola linea (il numero di linea è specificato dal valore di P2).

Modulo 4.3.2: Introduzione dei caratteri

Il primo compito importante del programma sarà quello di acquisire dati da tastiera e di elaborarli, cioè usarli per farci qualcosa. Bene, questo è anche il compito di questo primo modulo. La maggior parte delle linee che lo costituiscono hanno a che fare con l'introduzione di particolari caratteri di comando, che dicono al TEXTED di eseguire una data funzione, come cancellare un carattere, ma ci sono anche linee che già conoscete per averle usate nel capitolo sulla grafica, quando era necessario un certo spostamento del cursore.
La funzione globale del modulo è di consentirvi di introdurre fino a due righe di testo sulle linee 20 e 21 dello schermo. I moduli che verranno più avanti prenderanno queste righe e le ingloberanno nel testo principale.
Noterete inoltre che questo è uno dei pochi programmi del libro ad avere delle separazioni poste tra le linee per una maggiore chiarezza. Questo perché, anche se breve, il programma usa una tale quantità di nomi di variabili e brevi cicli, che

è molto difficile seguirne il flusso senza una certa guida visibile. Come i programma precedenti che sono stati presentati in questo modo, le linee di programma contenenti solo i due punti (:), volendo, possono essere tralasciate.

Modulo 4.3.2: Linee 12000-12430

```
12000 REM*******************************
12010 REM EDIT DI LINEA
12020 REM*******************************
12030 :
12040 :
12050 DO
12060 :
12070 :
12080 T$="" : DO WHILE T$=""
12090 CH=PEEK(FNA(P)) : POKE FNA(P)-1024,14 :
POKE FNA(P),160
12100 FOR TT=1 TO 5 : NEXT TT
12110 POKE FNA(P),CH
12120 GET T$
12130 LOOP
12140 :
12150 :
12160 IF T$=CHR$(13) OR (LEN(A$)>80 AND T$="
") THEN GOSUB 13000
12170 IF T$="↑" THEN GOSUB 15000
12180 :
12190 :
12200 DO WHILE T$="←"
12210 IF P<>0 THEN T=0 : ELSE T=LEN(A$)-1
12220 P=T
12230 EXIT : LOOP
12240 :
12250 :
12260 DO WHILE P>0 AND T$=CHR$(20)
12270 A$=LEFT$(A$,P-1)+MID$(A$,P+1)
12280 P=P-1
12290 EXIT : LOOP
12300 :
12310 :
12320 DO WHILE (T$=>" " AND T$<="]") OR
(T$=>"a" AND T$<="z")
```

```
12330 A$=LEFT$(A$,P)+T$+MID$(A$,P+1)
12340 P=P+1
12350 EXIT : LOOP
12360 :
12370 :
12380 IF T$="[CL]" AND P>0 THEN P=P-1
12390 IF T$="[CR]" AND P<LEN(A$)-1 THEN P=P+1
12400 :
12410 :
12420 CHAR,0,20,"[D BLU]" : PRINT A$
12430 LOOP
```

Commento

Linee 12080-12130: Un normale modulo per cursore lampeggiante (vedi capitolo 3), che lampeggia su spazio negativo sopra un carattere sulla linea 20 e 21. La posizione del cursore è indicata dal valore della variabile P.

Linea 12160: Questa linea seleziona la parte successiva del programma che incorpora ciò che viene introdotto nel testo principale. Ciò può essere fatto dall'utente con il tasto RETURN o automaticamente quando la lunghezza del testo supera gli 80 caratteri e viene introdotto uno spazio.

Linea 12170: Questo comando, il carattere ottenuto premendo SHIFT/0, chiama un modulo successivo che permette l'esecuzione di un gran numero di funzioni di editing sul testo principale.

Linee 12200-12230: Battendo SHIFT/= si ottiene come risultato lo spostamento del cursore all'inizio della riga o, se lo è già, alla fine.

Linee 12260-12290: Salvo il caso in cui il cursore lampeggiante sia posizionato sul primo carattere, premendo il tasto DEL si ottiene la cancellazione del carattere che si trova a sinistra del cursore lampeggiante. L'effetto è ottenuto ricostruendo la stringa A$ senza il carattere nella posizione P.

Linee 12320-12350: La linea 12320 riflette l'adozione di una tecnica molto importante che farete bene ad imparare. La maggior parte della gente ha difficoltà nel combinare fra loro più condizioni, usando AND e creando così condizioni complesse come:

IF I ≥ 2 AND I ≤ 6 THEN.....

Qui, l'azione da specificare sarà eseguita ogni volta che il valore di I è compreso fra 2 e 6. Cosa succede, però, se si vuole intraprendere una certa azione quando I è fra 2 e 6 o fra 10 e 15 e per nessun altro valore?
La risposta è che possiamo usare due serie di condizioni combinate con AND, collegate usando OR, come in:

IF (I ≥ 2 AND I ≤ 6) OR (I ≥ 10 AND I ≤ 15) THEN .....

L'importanza di quanto detto sta' nel fatto che i caratteri, che vogliamo usare per la normale introduzione di testo, di fatto si dividono in due parti: quelli dal codice di carattere 32 al 92, per il set di caratteri principale, e quelli dal 97 al 122, per il set di caratteri secondario che appare costituito da sole lettere maiuscole, quando il C16 viene portato nel modo di funzionamento a lettere minuscole, e da caratteri grafici, quando si trova nel modo a lettere maiuscole. Fra questi due gruppi, si trovano le due frecce di comando, che non vogliamo siano riprodotte dal programma TEXTED, e un carattere grafico. La combinazione delle due condizioni sulla linea 12320 permette la stampa di qualsiasi carattere riproducibile, esclusi i tre caratteri che hanno il codice da 94 a 96. Tutti i caratteri che superano la prova vengono inseriti in A$ e compaiono sullo schermo.

Linee 12380-12390: Lo spostamento del cursore lampeggiante a destra o a sinistra è ottenuto modificando il valore di P in base al tasto di spostamento del cursore usato. Notate in che modo, come nel capitolo dedicato alla grafica, la definizione di un nostro cursore ci permette di accettare solo ciò che vogliamo dai tasti di controllo del cursore come le frecce, l'inserimento, la cancellazione e così via. Nessuno di questi tasti ha un effetto automatico, dato che tutti si trovano fuori dal campo dei caratteri di stampa normali che vengono accettati dal ciclo precedente.
Possiamo però usarli quasi come tasti funzione, definendo l'effetto che vogliamo ottenere premendoli, salvo il tasto RUN/STOP che, ovviamente, dovrà sempre fermare l'esecuzione del programma.

Verifica

Come nel programma precedente, avrete bisogno di iniziare ad inserire linee del ciclo di controllo, quindi battete:

```
10050 IF IN=0 THEN GOSUB 11000
10070 GOSUB 12000
10100 END
```

e poi eseguite il programma. Dovreste vedere comparire un cursore lampeggiante all'inizio della linea sulla metà inferiore dello schermo. Ora cominciate a battere, ricordando che ciò che battete deve comparire sullo schermo. Dovreste inoltre essere in grado di cancellare usando il tasto DEL, spostare il cursore lampeggiante su tutto ciò che avete già battuto o farlo saltare qua e là dall'inizio alla fine usando SHIFT/=. Premendo RETURN, introducendo più di due linee di caratteri seguiti da uno spazio, dovreste provocare l'interruzione del programma con la comparsa del messaggio di errore UNDEFINED LINE.

Modulo 4.3.3: Inserimento di materiale nel blocco di testo principale

I prossimi due moduli lavorano insieme per permettere l'introduzione delle righe in fondo allo schermo e il loro conglobamento nel testo principale in TEXT$, che può contenere fino a 100 righe di 40 caratteri ciascuna. Il lavoro in sé è eseguito

da questo modulo, ma i risultati si vedranno solo dopo l'introduzione del prossimo che visualizza una parte di testo.

Modulo 4.3.3: Linee 13000-13310

```
13000 REM*******************************
13010 REM INSERIMENTO LINEA
13020 REM*******************************
13030 IF A$="* " THEN A$=FILL$
13040 X=0
13050 DO UNTIL A$=""
13060 :
13070 :
13080 DO WHILE LEN(A$)<42
13090 TT$(X)=LEFT$(A$,LEN(A$)-1)
13100 A$=""
13110 EXIT : LOOP
13120 :
13130 :
13140 DO WHILE LEN(A$)>41
13150 FOR I=41 TO 1 STEP-1
13160 IF MID$(A$,I,1)<>" " THEN NEXT I : I=41
13170 TT$(X)=LEFT$(A$,I-1)
13180 A$=MID$(A$,I+1)
13190 X=X+1
13200 LOOP
13210 LOOP
13220 X=X+1
13230 :
13240 :
13250 FOR I=LL+X TO PL+X STEP-1
13260 TEXT$(I)=TEXT$(I-X)
13270 NEXT I
13280 FOR I=0 TO X-1
13290 TEXT$(PL+I)=TT$(I)
13300 NEXT I
13310 A$=" " : P=0 : LL=LL+X : PL=PL+X
```

Commento

Linea 13030: Questa linea viene inserita per semplificare l'uscita del testo su una stampante. La sua azione può far sorgere qualche problema quando si

deve stampre una parola o una serie di parole sul lato destro della pagina. Poiché lo schermo è costituito da righe di 40 caratteri ciascuna, mentre qualsiasi stampante standard lavora con righe di 80, quando si deve stampare, il programma TEXTED usa due righe di testo su schermo per ogni riga stampata. L'introduzione di testo sul lato destro dello schermo può dare come risultato o la sua stampa in mezzo alla pagina o la sua aggiunta alla fine della riga precedente. Un metodo molto laborioso per ovviare a tutto questo è quello di introdurre prima una riga vuota, premendo RETURN se non c'è testo da introdurre. Ciò provoca la stampa di una riga vuota e garantisce che la riga successiva stampata inizi sul lato sinistro della pagina. A questo punto si batterà una riga di spazi completa, seguita da un'altra riga di spazi chiusa dalla parola da collocare a destra. La stampante stamperà tutti gli spazi e la frase finale comparirà sul lato destro del foglio. Nell'ultima frase si può anche fare a meno della riga piena di spazi e introdurne una con un unico asterisco. Questo modulo la trasferisce automaticamente in FILL$, definita con 40 spazi nel modulo di inizializzazione.

Linea 13040: X è la variabile che verrà usata per individuare quante righe sono necessarie per il nuovo testo da aggiungere.

Linee 13050 e 13210: Tutte le righe di caratteri aggiunte al testo principale, vengono ricavate da A$, che contiene il nuovo testo introdotto per mezzo del modulo precedente. Questo processo continua finché in A$ non c'è rimasto più niente.

Linee 13080-13110: Il nuovo testo viene prima di tutto collocato nel vettore provvisorio TT$. Se ci sono 41 caratteri o meno (cioè < 42), compreso lo spazio, che è sempre lì alla fine, il nuovo testo potrà andare in una riga da 40 caratteri. Tutto ciò che occorre fare è trasferirla e poi cancellare A

Linee 13140-13200: Se ci sono più di 41 caratteri e il nuovo testo non può stare quindi su una sola riga, occorre tornare indietro a partire dal carattere 41 fino al primo spazio che indica la fine della parola. Si può quindi trasferire la parte di A$ fino a questo punto, in modo che non ci siano parole spezzate in due. Questa parte di testo viene poi tolta dalla parte iniziale di A$ e viene di nuovo eseguito il ciclo principale esattamente nello stesso modo, salvo il fatto che il testo successivo viene posto in un'altra linea di TT$.

Linee 13250-13330 La variabile PL registra il punto (in inglese "Place") in cui vanno inserite le nuove linee nel testo principale (essa parte normalmente da zero, quando non c'è testo, e di solito rimane in fondo a ciò che è stato introdotto). I moduli successivi permetterrano all'utente di spostare il punto di inserimento in su' e in giù lungo tutto il testo principale. Il primo dei due cicli di questa parte inizia facendo posto al numero di righe nuove (memorizzato in X) nel punto in cui deve avvenire l'inserimento. Ciò viene fatto spostando tutto dal punto di inserimento fino all'ultima riga (LL), in sù per il numero di nuove righe, rappresentato da X. Il secondo ciclo copia il contenuto di TT$ nello spazio creato.

Verifica

Battete:

```
14130 RETURN
```

ed eseguite il programma. Non appena compare il cursore lampeggiante, battete:

```
AAA[RETURN]
BBB[RETURN]
CCC[RETURN]
DDD[RETURN]
```

e subito dopo premete STOP per porre fine al programma. Ora, nel modo diretto, battete:

```
FOR I = 0 TO 5:PRINT TEXT$(I):NEXT
```

A questo punto dovreste vedere una riga seghettata, seguita dalle quattro righe di testo che avete introdotto, seguite a loro volta da un'altra riga seghettata che indica la fine del testo.

Modulo 4.3.4: Visualizzazione del testo

Questo modulo visualizza quindici linee del blocco di testo principale.

Modulo 4.3.4: Linee 14000-14130

```
14000 REM*******************************
14010 REM VISUALIZZA SEZIONE TESTO
14020 REM*******************************
14030 SS=PL-7
14040 IF LL-PL<8 THEN SS=LL-15
14050 IF SS<0 THEN SS=0
14060 SCNCLR
14070 FOR I=SS TO SS+15
14080 PRINT "[BLK]";TEXT$(I);
14090 IF LEN(TEXT$(I))<40 THEN PRINT
14100 IF I=PL-1 THEN PRINT ">"
14110 NEXT I
14120 CHAR,0,20,"[D BLU]" : PRINT A$;"S"
14130 RETURN
```

Commento

Linee 14030-14050: La variabile SS rappresenta la riga iniziale della parte da stampare (SS), che viene calcolata in modo da essere normalmente otto righe prima del punto corrente di inserimento. Se quest'ultimo è molto vicino alla fine del testo, diciamo quasi in fondo, verranno visualizzate solo otto righe, in modo

che la linea 14040 sposti il punto di partenza più indietro. Infine, se ci sono meno di 15 righe o se il punto di inserimento è all'inizio, il punto di partenza ora sarà prima dell'inizio del testo, quindi viene portato a zero.

Linee 14070-14110: Le quindici righe sono ora stampate. Tutte terminano con un punto e virgola, per impedire che la posizione di stampa si sposti in giù di una riga. Questo perché, altrimenti, una riga di 40 caratteri piena sposterebbe il cursore sulla riga successiva, lasciando così una riga vuota.
Per le righe inferiori a quaranta caratteri, l'effetto del punto e virgola viene annullato da un comando PRINT nullo. L'unica altra rifinitura è rappresentata dal fatto che il punto di inserimento è segnato da un segno di maggiore (>) (tutte le funzioni di editing come la somma di righe, la loro copiatura o la loro cancellazione agiscono sulla riga che si trova sotto il marcatore).

Linea 14120: Per finire, poiché lo schermo è stato azzerato, tutto ciò che è contenuto in A$ viene ristampato in fondo allo schermo.

Verifica

Eseguite la stessa prova dell'ultimo modulo: la sola differenza, qui, è che invece di dover introdurre un comando speciale per vedere che cosa è stato battuto, basta premere RETURN perché tutte le righe siano collocate nel testo principale.

Modulo 4.3.5: Editing del testo principale

Dopo aver imparato ad editare il nuovo testo nel momento della sua battitura, abbiamo bisogno di poter disporre di un certo controllo anche sul testo principale in sé. Questo modulo da' per l'appunto all'utente la possibilità di spostare il punto di inserimento nel testo, di copiare linee a scelta del testo nella parte inferiore dello schermo, per poterle modificare, e di cancellare linee esistenti.

Modulo 4.3.5: Linee 15000-15330

```
15000 REM*********************************
15010 REM EDIT
15020 REM*********************************
15030 DO
15040 P2=PL-SS
15050 :
15060 :
15070 T1$="" : DO UNTIL T1$<>""
15080 POKE FNB(P2),32 : FOR I=1 TO 5 : NEXT
15090 POKE FNB(P2)-1024,8 : POKE FNB(P2),62 :
FOR I=1 TO 20 : NEXT
15100 GET T1$
15110 LOOP
15120 :
```

```
15130 :
15140 PL=PL+(T1$="[CU]")+10*(T1$="U") : IF
PL<1 THEN PL=1
15150 PL=PL-(T1$="[CD]")-10*(T1$="D") : IF
PL>LL THEN PL=LL
15160 IF T1$=CHR$(13) THEN EXIT
15170 :
15180 :
15190 DO WHILE PL<LL AND T1$=CHR$(20)
15200 FOR I=PL TO LL : TEXT$(I)=TEXT$(I+1) :
NEXT
15210 LL=LL-1
15220 EXIT : LOOP
15230 :
15240 :
15250 IF PL<LL AND T1$="C" THEN A$=TEXT$(PL)+"
"
15260 IF T1$="P" THEN GOSUB 17000
15270 IF T1$="S" THEN GOSUB 18000
15280 IF T1$="F" THEN GOSUB 16000
15290 :
15300 :
15310 GOSUB 14000
15320 LOOP
15330 RETURN
```

Commento

Linea 15040: Per gli scopi di questo modulo, il numero di linea del simbolo ">", che indica il punto di inserimento corrente nel testo, viene memorizzato nella variabile P2.

Linee 15070-15110: Il cursore > viene fatto lampeggiare ad indicare che il programma è nel modo edit.

Linee 15140-15150: Il cursore di editing può essere spostato in su' o in giù per tutto il testo, purché solo nel modo edit. I tasti con le frecce in su' e in giù muovono rispettivamente il cursore in sù o in giù di una riga, mentre i tasti U e D lo muovono nello stesso modo, ma di 10 righe per volta. Notate che in pratica, amenoché non si incontri l'inizio o la fine del file, non è il cursore a muoversi, ma il resto intorno a lui.

Linea 15160: La pressione del tasto RETURN pone termine al modo edit.

Linee 15190-15220: Premendo il tasto DEL nel modo edit si ottiene la cancellazione della riga di testo sotto il cursore di edit.

Linea 15250: Il tasto C, premuto nel modo edit, copia in fondo allo schermo la riga sotto il cursore, in modo che possa essere manipolata come nuovo testo.

Linea 15260: Il tasto P, premuto nel modo edit, richiama il modulo successivo che invia il testo alla stampante.

Linea 15270: Il tasto S, premuto nel modo edit, richiama il modulo successivo che salva il testo su disco.

Linea 15280: Il tasto F, premuto nel modo edit, richiama il modulo successivo che ripulisce il testo cercando di adattarlo il più possibile alla lunghezza delle righe.

Verifica

Con il modulo appena introdotto, eseguite la stessa prova del modulo precedente, in cui venivano inseriti quattro gruppi di lettere.

1) Ora battete SHIFT/0 e vedrete che il cursore comincia a lampeggiare.
2) Fate qualche esperimento muovendo il cursore in sù e in giù usando i tasti con le frecce. Se usate i tasti U e D vi accorgerete che essi vi portano o all'inizio o alla fine del testo dato che ciò che abbiamo a disposizione della prova è meno di 10 linee.
3) Spostate il cursore sulla riga immediatamente sopra l'ultima riga di testo, poi premete E e battete DDD per copiarla nella parte inferiore dello schermo.
4) Premete il tasto DEL e la linea DDD sparirà dal testo principale.
5) Spostate il cursore di edit in cima al testo.
6) Uscite dal modo edit premendo il tasto RETURN.
7) Quando il cursore lampeggiante ritorna in fondo allo schermo, premete di nuovo RETURN e vedrete ricomparire DDD all'inizio del testo.

Modulo 4.3.6: Formattazione del testo

Un'altra funzione molto utile è quella di ripulire il testo principale appena introdotto. Questo è necessario perché uno dei vantaggi principali dell'editing su testo, o del suo fratello maggiore, il word processor, è la capacità di prendere un testo già esistente e di staccare frasi o inserirne di nuove. Questo processo lascia molto spesso il testo in una forma grezza, poco piacevole, e per questa ragione, la maggior parte degli word processor, in grado di ripulire il testo automaticamente nello stesso momento in cui viene introdotto, permettono all'utente di usare una funzione di "formattazione", cioè di ricomposizione del testo secondo un formato prestabilito. A grandi linee, il compito di questa funzione consiste nel prendere ciascuna riga e stabilire se la prima parola della riga successiva potrebbe essere inserita alla fine (se sì, la parola viene spostata). Il risultato, anche se non vengono

inseriti degli spazi per fare in modo che tutte le righe finiscano nello stesso punto (questo processo è detto "giustificazione"), è la creazione di un testo più ordinato. Questo modulo esegue la formattazione sul testo conservato in TEXT$, usando la tecnica della suddivisione della stringa, di cui abbiamo parlato nel corso del primo capitolo.

Modulo 4.3.6: Linee 16000-16310

```
16000 REM*******************************
16010 REM FORMATO DELLA LINEA
16020 REM*******************************
16030 FOR I=1 TO LL-2
16040 DO WHILE TEXT$(I)<>"" AND TEXT$(I+1)<>""
16050 :
16060 :
16070 SP=40-LEN(TEXT$(I))
16080 WRD=INSTR(TEXT$(I+1)," ")
16090 IF WRD=0 THEN WRD=LEN(TEXT$(I+1))+1
16100 IF WRD>SP THEN EXIT
16110 :
16120 :
16130 DO WHILE SP=>WRD AND SP<=LEN(TEXT$(I+1))
16140 TEXT$(I)=TEXT$(I)+"
"+LEFT$(TEXT$(I+1),WRD-1)
16150 TEXT$(I+1)=MID$(TEXT$(I+1),WRD+1)
16160 EXIT : LOOP
16170 :
16180 :
16190 SP=40-LEN(TEXT$(I))
16200 DO WHILE LEN(TEXT$(I+1))<SP
16210 TEXT$(I)=TEXT$(I)+" "+TEXT$(I+1)
16220 FOR J=I+1 TO LL
16230 TEXT$(J)=TEXT$(J+1)
16240 NEXT J
16250 LL=LL-1 : PL=PL-1
16260 EXIT : LOOP
16270 :
16280 :
16290 LOOP
16300 NEXT I
16310 RETURN
```

Commento

Linee 16030 e 16300: Il processo di formattazione inizia con la prima riga e procede riga per riga su tutto il testo.

Linea 16040: L'operazione viene eseguita solo sulle righe che contengono qualcosa. Per esempio, alla fine di un paragrafo, ci può essere una riga con meno di 40 caratteri, ma in questo caso, non ci interessa che il nuovo paragrafo inizi dopo il suo ultimo carattere. La soluzione allora consiste nell'introdurre una nuova riga immediatamente dopo la fine del paragrafo. Il modulo di formattazione non aggiunge questa riga vuota alla fine di quella precedente, né copia parole della riga successiva nella riga vuota.

Linea 16070: La variabile SP viene usata per memorizzare la quantità di spazio lasciata alla fine della riga analizzata al momento.

Linea 16080: WRD viene usata per memorizzare la lunghezza della prima parola della riga successiva, come indicato dalla presenza di uno spazio.

Linea 16090: Se nella riga successiva non ci sono spazi, a WRD viene assegnata una lunghezza pari a quella della riga completa.

Linea 16100: Se la lunghezza della prima parola della riga successiva è maggiore dello spazio disponibile alla fine della riga corrente, non ha alcun senso continuare con quest'ultima, quindi il ciclo DO si ferma e il ciclo FOR si sposta sulla riga di testo successiva.

Linea 16130-16160: Queste linee vengono chiamate al lavoro se alla fine della riga corrente c'è spazio sufficiente per la parola successiva, non per tutta la riga.

Linea 16190-16260: È molto probabile che la riga corrente abbia abbastanza spazio per contenere tutta la riga successiva. In questo caso, non si tratta solo di copiare quest'ultima in quella corrente, tutto il vettore deve essere spostato in modo da riempire lo spazio che si è creato, altrimenti il programma sarà reso confuso dall'imposizione, di cui abbiamo parlato, secondo la quale le righe vuote devono essere lasciate da sole.

Verifica

Eseguite la stessa prova di prima, introducendo i quattro gruppi di lettere. Quando sono tutti visualizzati come parte del testo principale, entrate nel modo edit, premendo SHIFT/0 e subito dopo il tasto F. Dopo una pausa (che sarà piuttosto lunga se avete battuto molte righe) tornerà di nuovo sullo schermo il testo principale, ma questa volta i quattro gruppi di lettere devono trovarsi insieme sulla stessa linea.

Modulo 4.3.7: Memorizzazione dati

Un modulo del tutto normale. La sola cosa da notare è l'uso del carattere "@" che precede le linee quando vengono salvate. Questo perché le righe vuote, rappresentate da stringhe vuote o nulla in TEXT$, vanno normalmente perse al

momento di essere richiamate da disco. Il simbolo "@", posto all'inizio di tutte le linee, sta' ad indicare che non ci sono linee vuote, e tutto ciò che occorre fare al momento del richiamo è togliere il simbolo da ogni linea. La procedura ha inoltre lo scopo di proteggere gli spazi all'inizio delle righe, che potrebbero anche essere spariti.

Modulo 4.3.7: Linee 18000-18320

```
18000 REM*******************************
18010 REM FILE DATI
18020 REM*******************************
18030 Q$="" : DO UNTIL Q$="S"
18040 INPUT "NOME DEL FILE:";FI$
18050 PRINT "[CD]IL FILE DA REGISTRARE E'"FI$
18060 INPUT "[CD]IL NOME E' ESATTO (S/N)";Q$
18070 LOOP
18080 FI$="@0:"+FI$+",S,W"
18090 OPEN 1,8,14,FI$ : PRINT#1,PL :
PRINT#1,LL
18100 FOR I=0 TO LL : FF$=""+TEXT$(I) :
PRINT#1,FF$ : NEXT I
18110 CLOSE 1 : RETURN
18120 :
18130 :
18140 Q$="" : DO UNTIL Q$="S"
18150 INPUT "NOME DEL FILE:";FI$
18160 PRINT "[CD]IL FILE DA CARICARE E' "FI$
18170 INPUT "[CD]IL NOME E' ESATTO (S/N)";Q$
18180 LOOP
18190 FI$=FI$+",S,R"
18200 OPEN 15,8,15,"I" : CLOSE 15
18210 OPEN 1,8,2,FI$ : INPUT#1,PL,LL
18220 FOR I=0 TO LL
18230 TEXT$(I)=""
18240 T$="" : DO UNTIL T$=CHR$(13)
18250 TEXT$(I)=TEXT$(I)+T$
18260 GET#1,T$
18270 LOOP
18280 NEXT I : CLOSE 1
13290 FOR I=0 TO LL
18300 IF TEXT$(I)<>"@" THEN
TEXT$(I)=MID$(TEXT$(I),2)
```

```
18310 IF TEXT$(I)="@" THENTEXT$(I)=""
18320 NEXT I : RETURN
```

Modulo 4.3.8: Stampa del testo

Se avete una stampante (e un editor di testi non avrebbe davvero alcuna utilità se non l'aveste), questo è il modulo che vi permette di prendere il testo composto sullo schermo e di trasferirlo su carta. Così come è configurato, esso permette di sfruttare la possibilità della stampante di creare linee di 80 caratteri ciascuna. Se volete però creare margini maggiori, per esempio per scrivere le vostre lettere, esso è strutturato in modo tale da consentire di usare le funzioni di memorizzazione e formattazione su lunghezze di riga diverse, ad esempio 35 invece di 40 caratteri. In questo modo si può ottenere una larghezza di stampa di 70 colonne (o caratteri per riga) invece di 80.

Modulo 4.3.8: Linee 17000-17260

```
17000 REM*******************************
17010 REM USCITA SU STAMPANTE
17020 REM*******************************
17030 OPEN 7,4,7 : PRINT#7 : CLOSE 7
17040 OPEN 1,4
17050 X=1
17060 :
17070 :
17080 DO WHILE X<LL
17090 DO
17100 :
17110 :
17120 IF TEXT$(X)="" THEN PRINT#1,"" : X=X+1 :
EXIT
17130 PRINT#1,TEXT$(X);" ";
17140 X=X+1
17150 :
17160 :
17170 DO WHILE X<LL
17180 PRINT#1,TEXT$(X)
17190 IF TEXT$(X)="" THEN PRINT#1,""
17200 X=X+1
17210 EXIT : LOOP
17220 :
```

```
17230 :
17240 EXIT : LOOP
17250 LOOP
17260 PRINT#1,"" : CLOSE1 : RETURN
```

Commento

Linea 17030: Quasi tutte le stampanti compatibili con i computer Commodore hanno bisogno di sapere se devono stampare a lettere maiuscole o minuscole. Per la stampante Commodore che usiamo noi, questa linea serve a dire alla macchina di lavorare nel modo a lettere minuscole. Probabilmente non funzionerà sulla vostra stampante (essa è stata inserita solo per ricordarvi che dovete pensarci). Ciò che scriverete a questo punto dipende solo dalla vostra stampante e da quanto troverete sull'argomento nel suo manuale d'uso.

Linea 17040: Così come si aprono dei file per comunicare con il drive, così l'uscita viene inviata alla stampante aprendo un file ed usando l'istruzione PRINT# per inviarle le informazioni da stampare. Il numero di dispositivo per la vostra stampante sarà quasi inevitabilmente quattro, anche se in teoria potrebbe essere cinque o sei.

Linea 17050: La variabile X viene usata per registrare lo svilupparsi del modulo lungo le linee di TEXT$.

Linea 17120: Come nella formattazione, non viene fatto alcun tentativo per interferire con le righe vuote (ogni volta che se ne incontra una, si usa un'istruzione PRINT# vuota che sposta la stampante in giù di una riga).

Linea 17130: Se la riga non è vuota, viene stampata come prima metà di una riga di ottanta caratteri, seguita da uno spazio. Notate ii punto e virgola, che serve a garantire che i successivi caratteri stampati vengano immediamente dopo.

Linee 17170-17210: Queste linee provocano la stampa della seconda metà della riga. Se questa fosse vuota, verrebbe usata un'altra istruzione PRINT# che sposterebbe la stampante in giù di uno spazio.

Verifica

Se avete la stampante, controllate che sia collegata correttamente e che sia accesa, quindi battete un testo a piacere. Battete poi SHIFT/0, per entrare nel modo edit e infine premete P. Il testo dovrebbe essere generato sulla stampante. Se la prova ha successo, il programma può considerarsi pronto per essere usato. Per semplificarvi le cose, comunque, eccovi qui una tabella dei diversi comandi a un tasto che devono essere usati.

TEXTED; Tabella dei comandi a un tasto

Nel modo di introduzione del testo

RETURN	introduce nuove righe nel testo principale
SHIFT/=	sposta il cursore all'inizio o alla fine della riga
DEL	cancella il carattere a sinistra del cursore
FRECCE CURSORE	spostano il cursore a sinistra o a destra

Nel modo edit

RETURN	permette di uscire dal modo edit
FRECCE CURSORE	spostano il cursore in sù o in giù di una riga
U o D	spostano il cursore in sù o in giù di 10 righe
DEL	cancella la riga sotto il cursore
C	copia la riga sotto il cursore
P	stampa il testo principale corrente
S	salva su disco il testo principale corrente
F	formatta il testo principale

Modulo 4.3.9: Introduzione del modulo di controllo

E finalmente possiamo introdurre il modulo di controllo che ci permette di eseguire l'intero programma.

Modulo 4.3.9: Linee 10000-10100

```
10000 REM*******************************
10010 REM MODULO DI CONTROLLO
10020 REM*******************************
10030 TRAP 10080
10040 SCNCLR
10050 IF IN=0 THEN GOSUB 11000
10060 GOSUB 14000
10070 GOSUB 12000
10080 SCNCLR
10090 PRINT ERR$(ER),EL
10100 END
```

CAPITOLO 5

QUESTIONI DI SOLDI

In quest'ultimo capitolo rivolgeremo la nostra attenzione ad un aspetto importante che abbiamo finora trascurato, cioè il C16 e i calcoli con i valori monetari. É effettivamente un argomento che non può essere ignorato perché i microcomputer sanno trattarlo in modo davvero superbo. Gli importi elaborati sono spesso abbastanza contenuti o, se non lo sono, è molto difficile che siano trattati con un computer a basso costo come questo. Inoltre i calcoli da eseguire sono piuttosto elementari (i soldi di solito entrano ed escono, vengono sommati o sottratti fra loro).
Il vero vantaggio del microcomputer però non è rappresentato semplicemente dal fatto che sa trattare il denaro, anche il cervello umano ci riesce benissimo, ma soprattutto dalla sua capacità di abbinare operazioni come la memorizzazione delle informazioni, il loro richiamo rapido e la loro riproduzione stampata in forma facilmente comprensibile in funzione del tipo di trattamento che il denaro subisce.
I due programmi che vi presentiamo in questo capitolo semplificano questa abilità della macchina, il primo, permettendo all'utente di controllare il proprio conto bancario, il secondo, consentendogli di gestire la propria contabilità domestica.

Programma 5.1: BANKER

Lo scopo di questo programma è di permettere all'utente di tenere sotto controllo e sempre aggiornato il proprio conto corrente bancario, consentendogli di registra-

Figura 5.1: Tipico resoconto mensile

```
                    AGOSTO

SALDO MESE PRECEDENTE:   746500

GIORNO                                  IMPORTO   TOTALE


 1     SAL.                             835000   1581500
 3     LUCE                              23000   1558500
12     AUTO                             350000   1208500

PREMI UN TASTO PER CONTINUARE:
```

re le causali dei pagamenti, la data in cui sono stati effettuati e l'importo, oltre che di specificare non solo pagamenti singoli, ma addebiti o accrediti ricorrenti, indipendentemente dalla irregolarità del periodo. Il programma è stato studiato per gestire il conto corrente per il periodo di un anno, ma un numero molto elevato di transazioni e i limiti posti dalla memoria potrebbero ridurre la sua capacità a pochi mesi.

Modulo 5.1.1: Inizializzazione

Un modulo molto normale.

Modulo 5.1.1: Linee 10000-10160

```
10000 REM*******************************
10010 REM VARIABILI
10020 REM*******************************
10030 SCNCLR
10040 DO WHILE IN=0
10050 IN=1 : CR$=CHR$(13) : S$="      "
10060 DIM A$(99,1),A(99,1):A(0,1)=999
10070 RESTORE
10080 DIM MO$(11)
10090 FOR I=0 TO 11
10100 READ A$:MO$(I)=A$
10110 NEXT I
10120 DATA
GENNAIO,FEBBRAIO,MARZO,APRILE,MAGGIO,GIUGNO,LU
GLIO,AGOSTO
10130 DATA SETTEMBRE,OTTOBRE,NOVEMBRE,DICEMBRE
10140 INPUT "DEVI CARICARE DA DISCO (S/N):";Q$
10150 IF Q$="S" THEN GOSUB 15150
10160 LOOP
```

Commento

Linea 10060: Il vettore A$ viene usato per memorizzare le causali dei pagamenti e una stringa particolare, di cui parleremo più avanti, che registra i mesi in cui verrà fatto un certo pagamento. Il vettore numero A memorizza l'importo di ciascun pagamento e il giorno in cui va fatto. L'uso di A(0,1) posto a 999, un giorno inesistente, serve alla routine di ordinamento, che vedremo in seguito, per localizzare la fine del file.

Linee 10080-10130: Questo ciclo registra i nomi dei mesi dell'anno nel vettore MO$.

Modulo 5.1.2: Il menu del programma

Questo è un modulo menu molto normale con in più una piccola funzione che consente al programma di informare l'utente se gli verrà chiesto di generare i risultati prima dell'introduzione di qualsiasi dato.

Modulo 5.1.2: Linee 11000-11200

```
11000 REM*******************************
11010 REM MENU
11020 REM*******************************
11030 Z=0 : DO UNTIL Z=5
11040 COLOR 0,8 : SCNCLR : CHAR ,13,1,"[RVS
ON][RED]CONTO CORRENTE[GRN]": PRINT
11050 PRINT"[CD][CD][BLK]   COMANDI A
DISPOSIZIONE:"
11060 PRINT"[CD][GRN]      1)NUOVE OPERAZIONI"
11070 PRINT"[CD]      2)CONTROLLO/ELIMINA
OPERAZIONI"
11080 PRINT"[CD]      3)STAMPA RENDICONTO"
11090 PRINT"[CD]      4)REGISTRAZIONE DATI"
11100 PRINT"[CD]      5)STOP"
11110 INPUT "[CD][CD][BLK]  LA TUA SCELTA:";Z:
PRINT "";
11120 DO WHILE PA=0 AND (Z=2 OR Z=3)
11130 PRINT"[CD][CD][CD][CD][CD][CR]SPIACENTE
- MANCANO ANCORA I DATI.":GETKEYA$
11140 Z=0
11150 EXIT : LOOP
11160 ON Z GOSUB 12000,13000,14000,15000
11170 LOOP
11180 SCNCLR : CHAR,13,10,"[RVS ON][RED]CONTO
CORRENTE"
11190 CHAR,14,12,"[RVS ON]FINE  LAVORO"
11200 END
```

Modulo 5.1.3: Introduzione di nuove voci

Questo è un modulo di introduzione molto più complesso di quelli cui ci siamo abituati finora, per la semplice ragione che sono più complessi i dati in ingresso. Per ogni registrazione occorre conoscere cinque cose: se quello che viene genericamente chiamato pagamento è un accredito o un addebito, la causale del pagamento, il suo importo, i mesi e il giorno del mese in cui va effettuato.

Modulo 5.1.3: Linee 12000-12460

```
12000 REM*******************************
12010 REM INSERIMENTO NUOVE OPERAZIONI
12020 REM*******************************
12040 SCNCLR : CHAR,13,1,"[RED][RVS ON]NUOVE
OPERAZIONI[RVS OFF]" : PRINT
12050 PRINT"[CD][CD]      1) A CREDITO" :
PRINT"      2) A DEBITO"
12060 INPUT "[CD][BLK]  LA TUA
SCELTA:";CD:CD=CD-1
12070 INPUT "[CD][CD][GRN] NOME
DELL'OPERAZIONE:";Q$
12080 INPUT "[CD] IMPORTO:";Q
12090 EN=1 : DO UNTIL EN=0
12100 EN=0
12110 INPUT "[CD] MESI (ES. 01020410):";R$
12120 PRINT"[CD]";
12130 FOR I=1 TO LEN(R$) STEP 2
12140 M=VAL(MID$(R$,I,2))-1
12150 DO UNTIL M=>0 AND M<=11
12160 PRINT: PRINT"[CD]MESI INSERITI NON
CORRETTAMENTE          [CD][CD]"
12170 EN=1
12180 GETKEY A$
12190 EXIT : LOOP
12200 IF EN=0 THEN PRINTMO$(M);"/";
12210 NEXT I
12220 LOOP
12230 PRINT"
     "
12240 INPUT "[CD]GIORNO DELL'OPERAZIONE:";S
12250 INPUT "[CD][CD][RED]I DATI SONO ESATTI
(S/N):";T$
12260 IF T$<>"S" THEN RETURN
12270 PA=PA+1
12280 J=PA-1
12290 DO WHILE S<A(J,1)
12300 FOR K=0 TO 1
12310 A$(J+1,K)=A$(J,K)
12320 A(J+1,K)=A(J,K)
```

```
12330 NEXT K
12340 J=J-1 : IF J<0 THEN EXIT
12350 LOOP
12360 J=J+1
12370 A$(J,1)="000000000000"
12380 FOR I=1 TO LEN(R$) STEP 2
12390 M=VAL(MID$(R$,I,2))
12400
A$(J,1)=LEFT$(A$(J,1),M-1)+"1"+RIGHT$(A$(J,1),
12-M)
12410 NEXT I
12420 A$(J,0)=Q$
12430 A(J,0)=Q
12440 A(J,1)=S
12450 IF CD=1 THEN A(J,0)=A(J,0)*-1
12460 RETURN
```

Commento

Linee 12030 e 12260: L'intero ciclo che consente all'utente di confermare o annullare le informazioni fornite.

Linea 12060: Il programma ha chiaramente bisogno di sapere se la registrazione si riferisce ad un accredito o ad un addebito e la cosa viene risolta con la variabile CD (credito/debito).

Linee 12090-12220: I mesi in cui il pagamento deve essere effettuato sono introdotti nella forma di una stringa di numeri a due cifre, senza alcuna separazione fra loro. In questo modo se un pagamento trimestrale deve essere effettuato in febbraio, maggio, agosto e novembre, il valore introdotto si presenterà nella forma 02050811 (zero due per febbraio, zero cinque per maggio e così via). Il ciclo FOR che inizia con la linea di programma numero 12130 analizza la stringa per essere sicuro che i valori siano accettabili, informando eventualmente l'utente in caso di errore. Come ulteriore controllo, il ciclo stampa i nomi dei mesi specificati ricavandoli da MO$, in modo che l'utente possa stabilire che non solo sono accettabili ma sono anche quelli voluti. Nel caso in cui sia stato battuto un mese sbagliato, la routine che parte dalla linea 12090 viene ripetuta.

Linea 12220: A questo punto, le informazioni fornite sono state controllate e confermate dall'utente, perciò la variabile che registra il numero di voci del file, PA, viene incrementata di uno.

Linee 12290-12360: Lo scopo di questo ciclo è di porre la nuova voce nel giusto ordine di giorno. Invece di creare una doppia copia dei pagamenti che devono essere eseguiti in più mesi, il sistema adottato consiste nel memorizzare tutte le

registrazioni una volta sola, per ordine di giorno. Quando viene richiesto il resoconto di un certo mese, una parte del programma, che vedremo più avanti, passerà in rassegna tutte le registrazioni, controllandole una per una per vedere se si riferiscono a quel mese e possono quindi essere calcolate nella determinazione del saldo. Al momento dell'inserimento di una nuova voce, il ciclo che inizia dalla linea 12290 e termina con la linea 12350 parte con il giorno che ha il valore più alto (cioè quello fittizio di 999, inserito nel modulo di inizializzazione) e continua a scendere finché non trova il pagamento con un valore relativo al giorno inferiore al valore di S, cioè il valore della voce appena introdotta dall'utente. Se non trova la posizione corretta, sposta la voce appena esaminata in su di una posizione; in altre parole, scende lungo il file, spostando con sé una propria linea. Non appena trova la posizione giusta, la linea che ha con sé si trova già al suo posto senza aggiungere altro. Come ultima operazione, il valore di J, che registra la posizione della prima registrazione trovata, che ha un giorno di pagamento inferiore ad S, viene aumentata di uno in modo da puntare alla linea che il ciclo porta con sé.

Linee 12370-12410: Il compito successivo è quello di tradurre l'elenco dei mesi introdotto in un formato che possa essere facilmente analizzato dalle altre parti del programma. Il semplice espediente adottato consiste nell'usare una stringa di 12 zeri, che registra i mesi in cui il pagamento non deve essere effettuato, e poi un ciclo che cambia da zero a uno, per tutti i mesi in cui il pagamento va invece effettuato. In questo modo, nel caso del pagamento trimestrale di cui abbiamo parlato, la stringa si presenterà come 010010010010.

Linee 12420-12450: Le nuove informazioni vengono poste nei vettori principali. Se la variabile CD comunica che la voce è un debito, l'importo viene moltiplicato per meno 1, trasformandosi così in un numero negativo.

Verifica

Lanciate l'esecuzione del programma, scegliendo poi nel menu principale l'opzione uno.
Battete una nuova registrazione in questo modo:

Debito (opzione 2 al sollecito)

Nome: TEST
Importo: 100
Mesi: 02050811
Giorno: 15

Dopo una breve pausa dovreste ritornare al menu principale. Fermate l'esecuzione usando l'opzione cinque del menu. Poi battete:

```
PRINT A$(0,0),A$(0,1),A(0,0),A(0,1)
```

e dovreste ottenere come risultato i dati seguenti:

```
TEST 010010010010 - 100 15
```

Modulo 5.1.4: Visualizzazione del resoconto

Anche se ci sono ancora molti moduli che dobbiamo analizzare, vogliamo ora esaminare il compito conclusivo della parte principale di questo programma che consiste nel prendere le voci appena registrate usando il modulo precedente e nel comporle in modo ordinato in un resoconto mensile da ripetere per tutti i mesi dell'anno. Il resoconto conterrà il saldo, riportato di mese in mese, tutti i pagamenti di ogni mese e il saldo dopo ciascun pagamento.

Modulo 5.1.4: Linee 14000-14340

```
14000 REM*******************************
14010 REM COMPILAZIONE RENDICONTO
14020 REM*******************************
14030 SUM=0
14040 SCNCLR : CHAR,14,1,"[RED][RVS
ON]RENDICONTO[RVS OFF][BLK]" : PRINT
14050 Q=0 : DO UNTIL Q>0 AND Q<13
14060 INPUT "[CD]NUMERO DEL MESE DEL QUALE SI
DESIDERA    VEDERE IL RENDICONTO:";Q
14070 LOOP
14080 DO WHILE Q<>1
14090 FOR J=1 TO Q-1
14100 FOR I=0 TO PA-1
14110 IF MID$(A$(I,1),J,1)="1" THEN
SUM=SUM+A(I,0)
14120 NEXT I,J
14130 EXIT : LOOP
14140 SCNCLR : CHAR,15,1,"[RED]" :
PRINTMO$(Q-1)
14150 PRINT"[CD][BLK]SALDO MESE PRECEDENTE: ";
14160 IF SUM<0 THEN PRINT"[RED]";
14170 PRINT USING "####.##";ABS(SUM)
14175 PRINT"GIORNO"
14180 CHAR,24,4,"[GRN][RVS
ON]IMPORTO[CR][CR][CR]TOTALE[RVS OFF]" :
PRINT"[CD]"
14190 FOR I=0 TO PA-1
14200 DO WHILE MID$(A$(I,1),Q,1)="1"
14210 PRINT"[BLK]"; : PRINT USING "##";A(I,1);
: PRINT "    ";
14220 IF A(I,0)<0 THEN PRINT "[RED]";
```

```
14230 PRINT LEFT$(A$(I,0),15);
14240 PRINTTAB(23);
14250 PRINT USING "########";ABS(A(I,0));
14260 SUM=SUM+A(I,0)
14270 PRINT" [BLK]";
14280 IF SUM<0 THEN PRINT"[RED];
14290 PRINT USING "#####.##";ABS(SUM)
14300 EXIT : LOOP
14310 NEXT I
14320 PRINT"[CD]PREMI UN TASTO PER
CONTINUARE:"
14330 GETKEY A$
14340 RETURN
```

Commento

Linea 14030: La variabile SUM viene usata per contenere il saldo del conto (sia quello del mese precedente, sia quello da riportare nel mese successivo).

Linee 14080-14130: Salvo il caso in cui il saldo si riferisca al primo mese, nel qual caso mancherebbe il saldo precedente, questi due cicli analizzano l'intero elenco dei pagamenti relativi al mese precedente a quello per cui viene elaborato il resoconto. In questo modo ogni pagamento viene esaminato per vedere se è stato fatto in uno dei mesi precedenti, nel qual caso l'importo viene sommato al totale contenuto in SUM. Al termine dei due cicli, SUM contiene il saldo dopo tutti i pagamenti effettuati durante l'anno. (La tenuta di un saldo completo, che tenga conto anche dei soldi presenti all'inizio dell'anno, è possibile introducendo il saldo dalla fine dell'anno precedente come movimento di gennaio.)

Linee 14160-14170: Una cosa da osservare sulla stampa di SUM a questo punto è che, se l'importo è negativo, la linea 14060 cambia il colore, utilizzando per la stampa il colore rosso. Andando avanti vi accorgerete che questa tecnica verrà usata molto spesso. Notate inoltre l'uso del comando PRINT USING, che ci permette di imporre un formato standard al numero da stampare, garantendo così una presentazione ordinata del resoconto, con tutte le virgole decimali allineate.

Linee 14190-14310: Questo ciclo analizza l'intero elenco dei pagamenti, mentre il ciclo DO, in corrispondenza delle linee 14200 e 14300, sceglie solo quelli che hanno un 1 nella posizione più significativa della stringa che registra i mesi in cui deve essere effettuato il pagamento. Se un pagamento deve essere effettuato nel mese stabilito per il resoconto, il ciclo stampa il giorno, A(I,1), la causale, A$(I,0), l'importo, A(I,0) e infine il saldo prodotto, ottenuto aggiungendo l'importo all'ultimo totale di SUM. I debiti vengono sempre stampati in rosso, con la sola esclusione del giorno, per il quale viene usato il nero.
Lo schermo viene mantenuto suddiviso ordinatamente in colonne, anche se le

cifre possono essere di lunghezza diversa, usando TAB, che fa' partire ogni registrazione da una posizione standard dello schermo, e poi PRINT USING.

Verifica

Battete i dati di prova che avete usato nel modulo precedente poi, al ritorno al menu principale, scegliete l'opzione tre, quella che permette di stampare il resoconto.
Cominciate dal mese uno e, se siete soddisfatti del risultato, ritornate al menu principale e stampate il resoconto del mese successivo. A questo punto scoprirete che la maggior parte dei resoconti è vuota, dato che quasi tutti i mesi non hanno registrazioni. Solo i mesi di febbraio, maggio, agosto e novembre ne hanno una e la cosa risulterà visibile sullo schermo.

Modulo 5.1.5: Memorizzazione dati

Un modulo molto normale.

Modulo 5.1.5: Linee 15000-15260

```
15000 REM*******************************
15010 REM FILE DI DATI
15020 REM*******************************
15030 Q$="" : DO UNTIL Q$="S"
15040 INPUT "[CD]NOME DEL FILE DA
REGISTRARE:";FI$
15050 PRINT "[CD]IL NOME DEL FILE E' "FI$
15060 INPUT "[CD]IL DATO E' ESATTO (S/N):";Q$
15070 LOOP
15080 FI$="@0:"+FI$+",S,W"
15090 OPEN 1,8,2,FI$
15100 PRINT#1,PA
15110 FOR I=0 TO PA-1
15120
PRINT#1,A$(I,0);CR$;A$(I,1);CR$;A(I,0);CR$;A(I
,1)
15130 NEXT I
15140 CLOSE1 : RETURN
15150 Q$="" : DO UNTIL Q$="S"
15160 INPUT "[CD]NOME DEL FILE DA
CARICARE:";FI$
15170 PRINT"[CD]IL FILE DA CARICARE E' "FI$
15180 INPUT "[CD]IL NOME E' ESATTO (S/N):";Q$
15190 LOOP
```

```
15200 FI$=FI$+",S,R"
15210 OPEN 1,8,2,FI$
15220 INPUT#1,PA
15230 FOR I=0 TO PA-1 :
INPUT#1,A$(I,0),A$(I,1),A(I,0),A(I,1):NEXT
15240 A(PA,1)=999
15250 CLOSE1
15260 RETURN
```

Modulo 5.1.6: Modifica e cancellazione di registrazioni

Come nel programma NNUMBER, anche qui un modulo di modifica molto semplice che lavora però, questa volta, sulla base di un ciclo FOR che analizza tutte le registrazioni, una per una.

Modulo 5.1.6: Linee 13000-13250

```
13000 REM*******************************
13010 REM CONTROLLO/ELIMINA OPERAZIONI
13020 REM*******************************
13030 FOR I=0 TO PA-1
13040 COLOR 1,1 : SCNCLR
13050 PRINT"[CD]OPERAZIONE:";A$(I,0)
13060 PRINT"[CD]IMPORTO:";A(I,0)
13070 PRINT"[CD]MESE:";
13080 FOR J=1 TO 12
13090 IF MID$(A$(I,1),J,1)="1" THEN PRINT
MO$(J-1);"/";
13100 NEXT J : PRINT
13110 PRINT"[CD]GIORNO
DELL'OPERAZIONE:";A(I,1)
13120 PRINT"[CD][CD][CD][RED][RVS ON]COMANDI A
DISPOSIZIONE[RVS OFF][BLU]"
13130 PRINT"[CD][RVS ON]RETURN[RVS OFF]  -
OPERAZIONE SUCCESSIVA"
13140 PRINT"[RVS ON]ZZZ[RVS OFF]      - TORNA
AL MENU"
13150 PRINT"[RVS ON]0 (ZERO)[RVS OFF]-
CANCELLA L'OPERAZIONE"
13160 Q$="" : INPUT "[CD][GRN]LA TUA
SCELTA:";Q$
```

```
13170 DO WHILE Q$="0"
13180 FOR J=I TO PA-1
13190 FOR K=0 TO 1
13200 A$(J,K)=A$(J+1,K) : A(J,K)=A(J+1,K)
13210 NEXT K,J
13220 PA=PA-1
13230 EXIT : LOOP
13240 IF Q$="" THEN NEXT I
13250 RETURN
```

Verifica

Scegliete questa opzione dopo aver battuto un certo numero di dati e dopo averli memorizzati su disco. Dovreste essere in grado di scorrere lungo tutte le voci, cancellandole. Se queste funzioni sono utilizzabili significa che il programma è completo e perfettamente funzionante.

Programma 5.2.: ACCOUNTANT

Il secondo programma che vi presentiamo in questo capitolo è un po' più complesso di quello che abbiamo appena finito di analizzare. La sua funzione consiste nel tenere una serie molto elementare di conti, ciascuno diviso in debiti e crediti, o dare e avere, disponendoli nel formato tradizionale, con alcune voci isolate ad altre suddivise in gruppi che rappresentano diversi tipi di spesa.

Figura 5.2: Un breve documento contabile redatto con l'uso del programma ACCOUNTANT

```
SPESE MACCHINA
  BENZINA          50000
  OLIO              7500
                  ------
                              57500
CASA
  LUCE             36550
  GAS              12350
                  ------
                              48900
FERIE                        635000
                            -------
TOTALE                       741400
PREMI UN TASTO PER TORNARE AL MENU
```

Nello scrivere questo libro, siamo partiti dal presupposto che i nostri lettori non disponessero di una stampante e non avessero quindi molta memoria occupata dalle routine di stampa. Questo invece è fatto proprio per chi la stampante ce l'ha, e compatibile con il Commodore, e intende usarla. Anche sullo schermo, comunque, le presentazioni dei dati risulteranno chiare e facilmente comprensibili.

Modulo 5.2.1: Inizializzazione

Un modulo molto normale. Niente da dire.

Modulo 5.2.1: Linee 10000-10120

```
10000 REM*******************************
10010 REM INIZIALIZZAZIONE
10020 REM*******************************
10030 SCNCLR
10040 DO WHILE IN=0
10050 DIM A$(1,99),A(1,99)
10060 CR$=CHR$(13)
10070 C$="
    ":REM 40 SPAZI
10080 IN=1
10090 INPUT "DEVI CARICARE DA DISCO (S/N):";Q$
10110 IF Q$="S" THEN GOSUB 19200
10120 LOOP
```

Commento

Linee 10050: Le due colonne (debiti e crediti) di ogni conto, che contengono le causali di ciascun pagamento, sono memorizzati nei due "lati" dei vettori A e A$, in ciascuno dei quali possono essere eseguite fino a 100 registrazioni, per un totale di 200 per conto.

Modulo 5.2.2.: Il menu principale

Un modulo normale protetto però contro l'accesso a certe funzioni prima che i dati siano stati introdotti.

Modulo 5.2.2.: Linee 11000-11240

```
11000 REM*******************************
11010 REM MENU
11020 REM*******************************
11030 Z=0 : DO UNTIL Z=5
11040 COLOR 1,8 : SCNCLR
11050 CHAR,13,1,"[BLU][RVS ON]CONTABILITA'[RVS
```

```
OFF]" : PRINT
11060 PRINT"[CD][CD][RED]COMANDI A
DISPOSIZIONE:"
11070 PRINT"[CD][GRN]   1)INSERIMENTO NUOVE
RUBRICHE"
11080 PRINT"   2)MODIFICA/ELIMINA VOCI"
11090 PRINT"   3)STAMPA DEI CONTI"
11100 PRINT"   4)REGISTRAZIONE DATI"
11110 PRINT"   5)STOP"
11120 INPUT "[CD][BLU]LA TUA SCELTA:";Z
11130 IF Z<4 AND Z>0 THEN GOSUB 12000
11140 DO WHILE C(CD)=0 AND (Z=>2 AND Z<=3)
11150 PRINT"[CD][CD]NON E' STATO ANCORA
INSERITO ALCUN DATO"
11160 Z=0
11170 GETKEY A$
11180 EXIT : LOOP
11190 ON Z GOSUB 13000,16000,18000,13000,11210
11200 LOOP
11210 SCNCLR
11220 CHAR,13,10,"[RED][RVS
ON]CONTABILITA'[RVS OFF]"
11230 CHAR,13,12,"[BLU]FINE  LAVORO"
11240 END
```

Modulo 5.2.3: Credito o debito?

A differenza del programma BANKER, numerose parti di questo programma hanno bisogno di sapere se una certa registrazione è un debito o un credito. Questo è il motivo per cui, in un modulo a parte, viene inserita nel programma una routine per la richiesta di questa informazione all'utente.

Modulo 5.2.3: Linee 12000-12110

```
12000 REM*******************************
12010 REM ENTRATE E USCITE
12020 REM*******************************
12030 CD=-1 : DO UNTIL CD=0 OR CD=1
12040 CHAR ,1,15,"1) ENTRATE" : PRINT
12050 PRINT " 2) USCITE"
12060 INPUT "[CD][CD]LA TUA SCELTA:";CD
12070 CD=CD-1
```

```
12080 CD$="ENTRATE"
12090 IF CD=1 THEN CD$="USCITE"
12100 LOOP
12110 RETURN
```

Modulo 5.2.4: Il tipo di registrazione

Questo modulo è abbastanza elementare in sé, ma rappresenta anche uno dei motivi per cui questo programma è destinato a risultare molto più lungo di programmi come quello precedente. Scopo di questo modulo è di consentire all'utente di specificare una di queste tre cose:

1) Registrazione semplice. Tutto ciò che serve è la causale e l'importo. Quando viene stampato il conto, le singole registrazioni avranno le loro rispettive causali sulla sinistra e l'importo relativo nella colonna a destra.

2) Titolo di spesa. Si tratta in pratica di quell'elemento che accomuna certe registrazioni e che consente di specificarle nell'ambito del conto generale. Se voleste usare questo programma, ad esempio, per tenere i conti di casa, potreste aprire un conto AUTO, che raccolga piccoli conti di voci collegate come pneumatici, benzina, olio, meccanico e così via. In fase di stampa, il titolo di spesa verrà collocato sul lato sinistro, senza però avere il proprio importo corrispondente.

3) Sottotitolo. Come già detto al punto 2), ogni titolo di spesa può avere un elenco di voci, che formano un gruppo a sé. Nei conti, le causali dei sottotitoli saranno stampati sotto il loro relativo titolo, rientrati dal margine sinistro, mentre il loro importo risulterà a sinistra della colonna dei totali.

Modulo 5.2.4: Linee 13000-13130

```
13000 REM*******************************
13010 REM INSERIMENTO DATI
13020 REM*******************************
13030 SCNCLR
13040 CHAR,13,1,"[RED][RVS ON]NUOVE:"
13050 PRINT CD$
13060 PRINT"[BLU][CD][RVS OFF]COMANDI A
DISPOSIZIONE:"
13070 PRINT "[CD]  1)VOCE SINGOLA"
13080 PRINT "  2)CONTO PRINCIPALE"
13090 PRINT "  3)SOTTOCONTO"
13100 PRINT "  4)TORNA AL MENU"
13110 INPUT "[CD][PUR]LA TUA SCELTA:";TY
13120 ON TY GOSUB 14000,14000,15000
13130 RETURN
```

Verifica

Dopo aver introdotto tutte le parti del programma che non implicano calcoli di nessun genere, è consigliabile eseguire il programma e verificare la correttezza del menu. Se comunicate di voler introdurre una nuova registrazione, il programma dovrà rispondervi chiedendovi di specificare se si tratta di un debito o di un credito e di quale tipo (questo è per il momento tutto quello che potete fare). L'altra funzione che può essere utilizzata è la numero cinque che pone fine all'esecuzione. È lo stesso menu che dovrebbe impedirvi di accedere alle funzioni che consentono di modificare i dati o di stampare i conti, dato che non sono stati ancora introdotti dati.

Modulo 5.2.5: Ingresso di singole voci e di titoli di spesa

Esaminiamo ora i due moduli che si prendono cura, da un lato, dei sottotitoli e, dall'altro, delle singole voci o titoli. È importante, per poter capire le altre parti del programma, che cerchiate di seguire in che modo vengono memorizzate le singole voci e i caratteri indicatori che ne registrano il tipo.

Modulo 5.2.5: Linee 14000-14120

```
14000 REM*******************************
14010 REM VOCE SINGOLA/CONTO PRINCIPALE
14020 REM*******************************
14030 Q=0 : Q$="" : R$=""
14040 INPUT "[CD][BLU]NOME DELLA VOCE O DEL
CONTO:";Q$
14050 IF TY=1 THEN INPUT "[CD]IMPORTO DELLA
VOCE SINGOLA:";Q
14060 INPUT "[CD][RED]I DATI SONO ESATTI
(S/N):";R$
14070 IF R$<>"S" THEN PRINT "[CD]OPERAZIONE
ANNULLATA" : GETKEY A$ : RETURN
14080 IF TY=1 THEN Q$="%"+Q$ : ELSE Q$="*"+Q$
14090 A$(CD,C(CD))=Q$
14100 A(CD,C(CD))=Q
14110 C(CD)=C(CD)+1
14120 RETURN
```

Commento

Linea 14050: Come abbiamo già detto nell'introduzione al modulo precedente, i titoli non hanno mai importi relativi, per cui questa linea accetta un importo solo per le singole voci.

Linea 14080: A parte i settori crediti e debiti dei vettori, non esistono aree di

memoria separate per i diversi tipi di voci. I prossimi moduli stabiliranno il tipo di voce semplicemente guardando uno speciale carattere indicatore aggiunto all'inizio del suo nome e che può essere il segno di percentuale (%) per la singola voce, e l'asterisco (*) per il titolo.

Linee 14090-14120: Se avete seguito passo passo il nostro discorso, sapete già che cos'è la variabile CD e sapete anche che serve ad indicare se una certa voce è un debito o un credito. Qui, essa è usata per decidere quale "lato" dei vettori A e A$ deve contenere la registrazione corrente. Per quello che ci interessa, abbiamo bisogno di conservare il dato relativo al numero dei debiti e dei crediti, dato che saranno di solito diversi. Per questo viene usato il vettore C, che non è stato dichiarato nel modulo di inizializzazione dato che ha due soli elementi, C(O) e C(1), corrispondenti ai crediti e ai debiti dei vettori principali. Ancora una volta, il valore di CD viene usato per indicare quale dei due elementi di E va usato. Possiamo così vedere che quando si scrive:

```
A(CD,C(CD))
1 2 3
```

ciò che si intende è:

1) Un elemento del vettore numerico A
2) Sul lato indicato dal valore di CD, cioè credito o debito
3) Il primo elemento vuoto su quel lato, determinato in base a ciò che è già memorizzato.

Verifica

Eseguite il programma e scegliete la nuova opzione di ingresso dati. Specificate che intendete introdurre un titolo sul lato dei crediti, quindi scegliete come nome della registrazione le parole PROVA TIT (il programma non dovrebbe chiedervi alcun valore).
Scegliete ancora la nuova opzione e specificate questa volta una singola voce sul lato dei crediti, con il nome PROVA e il valore 100. Per finire, ripetete questa seconda parte della prova, specificando però il lato dei debiti.
Fermate l'esecuzione selezionandola nel menu principale e poi, nel modo diretto, battete:

```
PRINT A(0,0),A(1,0),A$(0,0),A$(1,0)[RETURN]
```

e dovreste vedere comparire sullo schermo questi dati:

```
0 0 *PROVA TIT *PROVA TIT
```

Ora eseguite la stessa procedura per la linea 1 dei vettori, ad esempio A(0,1) ecc.. e dovreste poter leggere:

```
100 100 %PROVA %PROVA
```

Infine generate il valore di C(O) e C(1) (entrambi devono essere uguali a due).

Modulo 5.2.6: Introduzione di un sottotitolo

Introdurre un nuovo sottotitolo non è così semplice come introdurre una singola voce. Per ogni nuovo sottotitolo, è infatti necessario eseguire un controllo per verificare la presenza del titolo e per eseguire il corretto abbinamento, dato che non avrebbe nessun senso aggiungere il sottotitolo a caso in fondo alle registrazioni effettuate.

Modulo 5.2.6: Linee 15000-15210

```
15000 REM*******************************
15010 REM SOTTOCONTO
15020 REM*******************************
15030 INPUT "[CD][BLU]CONTO PRINCIPALE:";Q$
15040 Q$="*"+Q$
15050 PL=-1 : FOR I=0 TO C(CD)-1
15060 IF A$(CD,I)=Q$ THEN PL=I+1
15070 NEXT I
15080 IF PL=-1 THEN PRINT "IL CONTO NON
ESISTE!" : GETKEY A$ : RETURN
15090 INPUT "[CD][BLK]NOME DEL SOTTOCONTO:";Q$
15100 INPUT "[CD]IMPORTO:";Q
15110 R$="" : INPUT "[CD][RED]I DATI SONO
ESATTI (S/N):";R$
15120 IF R$<>"S" THEN PRINT "[CD]OPERAZIONE
ANNULLATA" : GETKEY A$ : RETURN
15130 Q$="$"+Q$
15140 FOR I=C(CD)+1 TO PL+1 STEP-1
15150 A$(CD,I)=A$(CD,I-1)
15160 A(CD,I)=A(CD,I-1)
15170 NEXT I
15180 A$(CD,PL)=Q$
15190 A(CD,PL)=Q
15200 C(CD)=C(CD)+1
15210 RETURN
```

Commento

Linee 15030-15080: Il nome del titolo è necessario. Poi viene fatto un controllo delle voci già presenti nel file, per vedere se l'intestazione esiste realmente. Se non esiste, viene generato un messaggio di errore e l'esecuzione ritorna al menu.

Linee 15140-15200: Come già detto in precedenza, il problema del sottotitolo sta' nel fatto che esso deve comparire nei conti principali come parte di un gruppo

stampato sotto il titolo. Per ottenere questo effetto in modo semplice, il metodo più efficace consiste nel memorizzarlo nel file vicino al relativo titolo. La posizione della prima voce immediatamente successiva al titolo è già stata individuata dal ciclo FOR alla linea 15050, in questo modo tutto ciò che occorre fare è spostare in sù tutte le voci da quel punto e porre una nuova voce nel vettore al punto giusto.

Verifica

Introducete le registrazioni proposte per la verifica del modulo 5.2.5, poi richiamate il nuovo modulo di introduzione e collocate un nuovo sottotitolo sul lato dei crediti, scegliendo il nome PROVA SOT, con valore 200. Ripetete poi la stessa cosa, ma questa volta supponendo un debito.
Nel modo diretto, introducete queste informazioni:

```
FOR I = 0 TO 2:PRINT A(0,I),A(1,I),A$(0,I),A$(1,I):NEXT I0IRETURN]
```

e dovreste poter leggere quanto segue:

```
0     0     *PROVA TIT     *PROVA TIT
200   200   $PROVA SOT     $PROVA SOT
100   100   %PROVA         %PROVA
```

Stampati, i valori di C(0) e C(1) dovrebbero risultare entrambi uguali a 3.

Modulo 5.2.7: File di dati

Poiché i dati per il programma ACCOUNTANT sono molto complessi, è probabilmente consigliabile introdurre il modulo del file dati a questo punto per non dover aver bisogno della continua ribattitura di dati durante le prove. Una volta che il modulo è stato introdotto, battete e salvate i dati specificati per la prova del modulo precedente.

Modulo 5.2.7: Linee 19000-19330

```
19000 REM*******************************
19010 REM REGISTRAZIONE DATI
19020 REM*******************************
19030 R$="" : DO UNTIL R$="S"
19040 INPUT "[CD]NOME DEL FILE DA
REGISTRARE:";FI$
19050 PRINT "[CD]IL NOME DEL FILE E' "FI$
19060 INPUT "[CD]IL NOME E' ESATTO (S/N):";R$
19070 LOOP
19080 FI$="@0:"+FI$+",S,W"
19090 OPEN 1,8,2,FI$
19100 FOR I= 0 TO 1
19110 PRINT#1,C(I)
19120 DO WHILE C(I)<>0
```

```
19130 FOR J=0 TO C(I)-1
19140 T$=" "+A$(I,J)
19150 PRINT#1,T$;CR$;A(I,J)
19160 NEXT J
19170 EXIT : LOOP
19180 NEXT I
19190 CLOSE1 : RETURN
19200 R$="" : DO UNTIL R$="S"
19210 INPUT "[CD]NOME DEL FILE DA
CARICARE:";FI$
19220 PRINT "[CD]IL NOME DEL FILE E' "FI$
19230 INPUT "[CD]IL NOME E' ESATTO (S/N):";R$
19240 LOOP
19250 OPEN 1,8,2,FI$
19260 FOR I=0 TO 1 : INPUT#1,C(I)
19270 DO WHILE C(I)<>0
19280 FOR J=0 TO C(I)-1
19290 INPUT#1,A$(I,J),A(I,J)
19300 NEXT J
19310 EXIT : LOOP
19320 NEXT I
19330 CLOSE1 : RETURN
```

Modulo 5.2.8: Modifica delle voci

Questo è un modulo con alcune funzioni in più che tengono conto del fatto che alcune voci non stanno mai sole, ma vengono trattate come parte di un gruppo omegeneo sotto un'intestazione comune.

Modulo 5.2.8: Linee 16000-16290

```
16000 REM*******************************
16010 REM MODIFICA ED ELIMINA
16020 REM*******************************
16030 FOR I=0 TO C(CD)-1
16040 Q$="" : DO UNTIL Q$="1"
16050 SCNCLR
16060 CHAR,10,1,"[RED][RVS ON]MODIFICA O
ELIMINA[RVS OFF]": PRINT
16070 IF LEFT$(A$(CD,I),1)<>"$" THEN
PRINT"[CD][CD]";MID$(A$(CD,I),2);
16080 IF LEFT$(A$(CD,I),1)="*" THEN
```

```
HH$=MID$(A$(CD,I),2) : PRINT
16090 IF LEFT$(A$(CD,I),1)="$" THEN
PRINT"[CD][CD]";HH$ : PRINT
"[CD]";MID$(A$(CD,I),2);
16100 DO WHILE A(CD,I)<>0
16110 PRINT TAB(28); : PRINT USING
"##########";A(CD,I)
16120 EXIT : LOOP
16130 CHAR,1,8,"[RED]COMANDI A DISPOSIZIONE:"
: PRINT
16140 PRINT"[CD][GRN]  1 - VOCE SEGUENTE"
16150 PRINT"  2 - MODIFICA IMPORTO"
16160 PRINT"  3 - TORNA AL MENU"
16170 PRINT"  4 - CANCELLA LA VOCE"
16180 PRINT"[CD][CD][PUR]LA TUA SCELTA:?"
16190 GETKEY Q$
16200 IF Q$="4" THEN GOSUB 17000 : RETURN
16210 IF Q$="3" THEN RETURN
16220 DO WHILE Q$="2" AND
LEFT$(A$(CD,I),1)<>"*"
16230 Q=0 : INPUT "[CD][CD][BLK]SOMMA DA
AGGIUNGERE:";Q
16240 R$="" : INPUT "[CD][RED]L'IMPORTO E'
ESATTO (S/N):";R$
16250 IF R$="S" THEN A(CD,I)=A(CD,I)+Q
16260 EXIT : LOOP
16270 LOOP
16280 NEXT I
16290 RETURN
```

Commento

Linee 16070-16090: Se la voce richiamata dal file è una voce singola, viene riprodotta, anche se senza il carattere indicatore che viene aggiunto all'inizio del nome. Se invece è un titolo, non viene solo riprodotto, ma il suo nome viene memorizzato in HH$ in modo che possa essere visualizzato o stampato sopra tutti i sottotitoli che seguiranno.

Linee 16220-16260: Oltre che cancellare le voci, è possibile apportare delle modifiche al valore associato al titolo. L'operazione viene eseguita introducendo l'importo positivo o negativo con il quale deve essere modificato il valore di una voce (non si tratta di un valore assoluto che deve essere acquisito dalla voce).

Il vantaggio di una tale scelta sta' nel fatto che la maggior parte delle modifiche determinerà la necessità di aggiungere importi alle voci esistenti a mano a mano che vengono effettuate nuove spese o vengono incassati crediti sotto voci che già esistono. Così, se si spendono 100 lire per riparazioni alla macchina (una voce che è già inclusa sotto un titolo di spesa), tutto ciò che occorre fare è scorrere lungo i dati del file fino al titolo e poi introdurre il valore 100.

Verifica

Battete il comando RUN e richiamate i dati che avete già memorizzato su disco. Ora selezionate l'opzione due del menu e verificate di poter scorrere lungo le tre voci e di ritornare al menu. Scegliete un'altra volta l'opzione due, questa volta cercando di sommare o sottrarre un valore dai due totali precedentemente introdotti. Scorrendo di nuovo lungo i dati, dovreste poter capire se la prova ha avuto o meno successo.

Modulo 5.2.9: Cancellazione di registrazioni

Un'ultima funzione da aggiungere al nostro programma è quella della cancellazione delle registrazioni già introdotte. In questo programma, il modulo di cancellazione è più complesso di quello degli esempi precedenti. Questo perché, intorno ai titoli, esistono dei gruppi di voci. Se cancellare una voce non è infatti molto difficile, così come relativamente semplice è cancellare un sottotitolo, cosa succede se proviamo a cancellare un titolo di spesa? È ovvio che per farlo dobbiamo cancellare anche tutti i relativi sottotitoli, altrimenti il programma tratterebbe delle entità, che per loro natura sono aggregate sotto una specifica comune senza che questa sia indicata, e il conto perderebbe quindi di senso.

Modulo 5.2.9: Linee 17000-17140

```
17000 REM*******************************
17010 REM ELIMINAZIONE
17020 REM*******************************
17030 PL=1 : GR=1
17040 DO WHILE LEFT$(A$(CD,PL),1)="*"
17050 DO UNTIL LEFT$(A$(CD,PL+GR),1)<>"$"
17060 GR=GR+1
17070 LOOP
17080 EXIT : LOOP
17090 FOR K=PL TO C(CD)-GR-1
17100 A(CD,K)=A(CD,K+GR)
17110 A$(CD,K)=A$(CD,K+GR)
17120 NEXT K
17130 C(CD)=C(CD)-GR
17140 RETURN
```

Commento

Linea 17030: Il punto in cui deve aver luogo la cancellazione viene comunicato dal modulo precedente nella forma della variabile I, che, per gli scopi di questo modulo, viene trasferita in PL. La variabile GR (che sta' per GRuppo) registra il numero delle voci da cancellare. All'inizio assume il valore 1 e questo valore viene incrementato solo se la voce da cancellare è un titolo con i suoi sottotitoli.

Linee 17040-17080: Questi due cicli vengono attivati solo se la voce che deve essere cancellata è un titolo. Il ciclo più interno analizza tutti gli ingressi successivi, contando quante di quelle voci che seguono sono precedute dal simbolo del dollaro, che sta' ad indicare che si tratta di voci associate ad un titolo. Il risultato del conteggio è tenuto nella variabile GR.

Linee 17090-17120: Un normale ciclo per la cancellazione di una registrazione. La differenza qui sta' nel fatto che invece di copiare la voce X nello spazio X-1 e quindi invece di copiare ogni elemento più in giù di un posto, le voci vengono trasferite di GR posti, e GR voci vengono cancellate (GR è il numero di voci del gruppo costruito intorno ad un titolo).

Verifica

Seguendo la procedura di verifica del modulo precedente, dovreste non solo essere in grado di scorrere lungo le voci e di modificarle, ma dovreste anche poterle cancellare. Se cancellate la voce di nome PROVA TIT, con essa dovrebbe sparire anche quella chiamata PROVA SOT.

Modulo 5.2.10: Visualizzazione dei conti

Dopo tutta questa lunga preparazione, ecco l'unico modulo che da' un senso a questo programma, visualizzando il conto nella sua forma definitiva. Come il modulo equivalente del programma BANKER, sembra piuttosto complesso, ma dopo che avrete visto una riproduzione su schermo, capirete che tutto è disposto secondo un preciso significato.

Modulo 5.2.10: Linee 18000-18300

```
18000 REM*******************************
18010 REM STAMPA CONTI
18020 REM*******************************
18030 TT=0 : SS=0
18040 SCNCLR
18050 CHAR,15,1,"[RVS ON][RED]" : PRINT
CD$;"[CD]"
18060 FOR I=0 TO C(CD)-1
18070 TT=TT+A(CD,I)
18090 COLOR 1,1
18100 IF LEFT$(A$(CD,I),1)="*" THEN PRINT
```

```
18110 IF LEFT$(A$(CD,I),1)="$" THEN
PRINT"[CR][CR]";
18120 PRINTMID$(A$(CD,I),2)
18130 DO WHILE LEFT$(A$(CD,I),1)<>"*"
18140 PRINT TAB(18);"[CU]";
18150 IF LEFT$(A$(CD,I),1)="%" THEN PRINT
TAB(29);
18160 PRINT USING "########";A(CD,I)
18170 IF LEFT$(A$(CD,I),1)="$" THEN
SS=SS+A(CD,I)
18180 EXIT : LOOP
18190 DO WHILE SS<>0 AND
LEFT$(A$(CD,I+1),1)<>"$"
18200 PRINT TAB(18);"---------"
18210 PRINT TAB(29);
18220 PRINT USING"########";SS
18225 SS=0
18230 EXIT : LOOP
18235 GETKEY T$
18240 NEXT I : PRINT TAB(29);"---------"
18250 PRINT"[RVS ON]TOTALE[RVS OFF]:";
18260 PRINT TAB(29);
18270 PRINT USING "########";TT
18280 PRINT"[CD][RED]PREMI UN TASTO PER
TORNARE AL MENU"
18290 GETKEY A$
18300 RETURN
```

Commento

Linea 18070: La variabile TT verrà usata per memorizzare il totale calcolato per il conto al momento della sua stampa.

Linee 18080-18090: Queste due linee cambiano il colore di stampa per avere righe nere e verdi alternate. La sola ragione di questa scelta è poter semplificare la lettura dei conti, avendo sulla stessa linea ma distanti le causali dei movimenti del conto e i loro relativi importi.

Linee 18100-18120: Queste linee stampano la causale. Per i titoli viene prima generata una linea vuota per separarli da ciò che li ha preceduti, mentre per i sottotitoli si ha un rientro di due spazi rispetto al margine sinistro.

Linee 18130-18180: Queste linee hanno a che fare con la stampa degli importi delle singole voci e dei sottotitoli. Questi ultimi verranno stampati a partire dalla diciottesima posizione della linea, mentre le singole voci a partire dalla ventinovesima. Se il modulo ha a che fare con una singola linea, il totale parziale del gruppo elaborato al momento viene memorizzato temporaneamente in SS.

Linee 18190-18230: Questo ciclo lavora solo quando un gruppo è in fase di elaborazione, come indicato dal fatto che SS non è uguale a zero, e la voce successiva non fa' parte del gruppo (cioè quest'ultimo è completo). L'effetto del ciclo è quello di stampare il totale relativo al gruppo nella colonna dei totali a partire dalla posizione 29.

Linea 18235: Viene stampata una voce per volta. Questo per evitare che la parte superiore del conto possa scorrere al di sopra dello schermo prima che possa essere letta. Quindi occorre premere un tasto per ogni nuova voce.

Verifica

Ricaricate i dati che avete precedentemente memorizzato su disco e scegliete nel menu principale l'opzione tre. Ricordate che, per ottenere la visualizzazione dell'intero conto, dovete premere un tasto per ogni voce (il conto non appare immediatamente per intero). Se la prova ha avuto successo, il programma può essere considerato pronto per essere usato.

SUNSHINE

Leggendo e "usando" questo libro vi troverete, alla fine, capaci di adottare e sviluppare con estrema facilità, per i vostri scopi, l'insieme delle subroutine contenute nel testo; questo grazie alla graduale crescita delle difficoltà e alle ottime spiegazioni che affiancano i programmi. Dopo aver creato agende computerizzate, sistemi di orologi, selezioni di immagini, motivi musicali a due voci, arriverete a gestire archivi e dizionari, ad elaborare fatture, listini di borsa... e anche il vostro computer "lavorerà" più volentieri con voi.

L. 29.000

COD 422B ISBN 88-7056-257-3